MASTER EN SCIENCES INFORMATIQUES

Un langage de haut niveau pour l'exploitation de bases de données CODASYL structure des données

Lan, Nguyen

Award date:
1979

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX - NAMUR
INSTITUT D'INFORMATIQUE

UN LANGAGE DE HAUT NIVEAU POUR L'EXPLOITATION DE BASES DE DONNEES CODASYL : STRUCTURE DES DONNEES

Promoteur : J.L. HAINAUT

NGUYEN LAN

Mémoire présenté en vue de l'obtention du grade de
LICENCIE ET MAITRE EN INFORMATIQUE

ANNEE ACADEMIQUE : 1978-1979

Je tiens à exprimer ma profonde gratitude à Mr. Jean-Luc HAINAUT pour l'intérêt porté à ce mémoire et l'aide constante qu'il m'a prodiguée lors de la rédaction. Ses nombreux conseils m'ont permis de clarifier les difficultés théoriques de cette étude et d'en réaliser une implémentation.

Ma gratitude va également à toutes les personnes qui, de près ou de loin, ont collaboré à la réalisation de ce mémoire.

TABLE DES MATIERES

3ème PARTIE : SCHEMA CONCEPTUEL

4ème PARTIE : REALISATION DU SCHEMA

°
° °

INTRODUCTION

L'étude des bases de données peut être décomposée en quatre niveaux :

- le niveau conceptuel ou fonctionnel
- le niveau des accès logiques
- le niveau des structures physiques
- le niveau externe

et les relations entre eux.
Les bases de données exigent des systèmes de gestion puissants et forts complexes et en même temps des représentations externes très simples, destinées aux utilisateurs.

La simplicité d'un modèle de BD (Bases de données) se traduit :

- au niveau des DDL (*) en proposant un langage de description simple, facile à comprendre;
- au niveau du DML (**) en créant des primitives puissantes mais faciles à manipuler.

Et tous les deux ne nécessitent pas une maîtrise de la structure physique d'implémentation des données ainsi que des mécanismes de gestion internes.

Les spécifications du CODASYL ont été proposées par le "Data Base Task Group" (DBTG), on peut dire que c'est le modèle CODASYL. D'autre part, le DBTG a défini les caractéristiques générales d'un Système de Gestion de Bases de Données (SGBD) associé. A l'heure actuelle, les propositions CODASUL sont réalisées sur plusieurs machines : UNIVAC, SIEMENS, IBM, PHILIPS, CDC, PDP ...
Le SGBD-CODASYL fournit des moyens de gestion de BD (Bases de Données) puissants et est commercialisé, par contre le modèle CODASYL est lié étroi-

---

(*) DDL : Langage de description de données
(**) DML : Langage de manipulation de données

tement aux structures physiques d'implémentation des données et il demande de la part de l'utilisateur une parfaite connaissance des particularités du SGBD réalisant ce modèle. D'autre part, les ordres du DML-CODASYL sont, de manière générale de type "ponctuel" dans le sens que des actions s'effectuent sur un seul article (ou enregistrement) à la fois.

Le modèle d'accès a été développé à l'Institut d'Informatique de Namur à partir des concepts communs des différents SGBD. Le SGBD-SPHYNX a été réalisé à partir des définitions de ce modèle d'accès dans le cadre d'un projet de recherche sur les modèles, langages et système pour la conception et l'exploitation de BD. Il n'est pas commercialisé.

Le DML du modèle d'accès associé au système SPHINX (DML-SPHINX) est un langage de manipulation de données de haut niveau. (On peut distinguer, d'une manière relative, 2 classes de DML :
- ceux dont les ordres s'effectuent sur un seul article;
- ceux dont les ordres s'effectuent sur un ensemble d'articles, notamment l'ordre d'accès (ceux qu'on appelle DML de haut niveau comme SEQUEL, SOCRATE, différents IQL)

Dans le cadre de ce projet, l'exploitation des BD est réalisée grâce au système SPHINX qui prend en charge des requêtes des différents programmes DML-SPHINX et c'est lui qui communique avec le système cible SESAM (construit par SIEMENS), ce dernier effectue des ordres directement sur les BD.

L'objectif de notre travail est de représenter le modèle CODASYL par un modèle plus simple évitant les inconvénients cités plus haut en vue de la construction d'un DML de type DML-SPHINX (ce DML et son compilateur sont étudiés dans le travail de Melle B. Rossel), ce qui permet d'exploiter des BD CODASYL par un tel DML (au lieu d'utiliser directement le DML-CODASYL)

La réduction des inconvénients et la conciliation des avantages du CODASYL et du SPHINX associé au modèle d'accès permet de réaliser la simplicité d'un modèle (d'une manière relative). Le fait que les spécifications CODASYL ont été implémentées par différents constructeurs assure de plus à un tel système, un certain degré de généralité.

Une telle exploitation des BD-CODASYL demande la réalisation d'un SGBD superposé sur celui de CODASYL. En effet, elle demande :

- une description d'une BD-CODASYL sous forme de celle du Modèle CODASYL Simplifié (MCS), cette description est destinée aux utilisateurs DML ainsi construit. (génération de listings DDL décrivant le modèle CODASYL simplifié (DDL-MCS))
- la génération des tables de ce compilateur DML. (Nous appelons DML-MCS ce langage de manipulation de données pour éviter la confusion).

Ces tables ont pour but :

1° d'établir une zone de communication entre un programme DML et le "User working aréa" (UWA), ce dernier est généré et géré par SGBD-CODASYL.

2° de produire les ordres DML-CODASYL correspondants.

Le modèle CODASYL simplifié est défini à partir :

- du modèle d'accès et du DML-SPHINX associé
- des principaux concepts de CODASYL.

Pour satisfaire les besoins du SGBD, nous allons construire une BD des schémas CODASYL et nous suivons une démarche classique d'analyse des systèmes d'information :

- une analyse conceptuelle des types de données CODASYL. Le rapport CODASYL-71 sert comme document de base de cette analyse, tandis que la réalisation de notre travail est effectuée sur DBMS-20 qui est de type CODASYL et implémenté sur DEC-20.
- une analyse des accès logiques des deux applications : génération des listings DDL du modèle CODASYL simplifié (DDL-MCS) et génération des tables pour compilateur DML-MCS
- une analyse organique du système DBMS-20.

Cet exposé comporte quatre parties :

Dans la première partie, nous faisons une étude théorique du modèle d'accès et de CODASYL. Nous en déduirons ensuite un modèle CODASYL simplifié.

La deuxième partie présente les deux DDL décrivant le modèle simplifié, l'un pour CODASYL et l'autre, avec quelques modifications, pour DBMS-20.

L'analyse conceptuelle des types de données CODASYL et celle de DBMS-20 font l'objet d'une troisème partie.

Enfin, la quatrième partie consiste à réaliser une base de données correspondant au schéma conceptuel ainsi construit et les trois applications :

- chargement de la BD;
- génération des listings DDL-MCS;
- génération des tables pour le compilateur DML-MCS.

°
° °

Ière PARTIE :

ETUDE THEORIQUE

CHAPITRE I : MODELE D'ACCES ET DPL-SPHINX

1.1. Introduction

Le Modèle d'Accès est un modèle de bases de données qui a été défini par l'équipe Grands Fichiers, de l'Institut d'Informatique de Namur.

Le modèle d'accès a été construit à partir d'un ensemble de notions rendant compte de la plupart des propriétés des types d'information et des primitives que l'on peut rencontrer dans les systèmes de gestion de bases de données (SGBD), en éliminant les détails techniques et certaines contraintes propres à tel ou tel système particulier.

Dans ce chapitre, nous rappelons brièvement les principaux concepts, les propriétés fondamentales des types d'informations et les primitives de manipulation des données du modèle d'accès ainsi que le DML-SPHINX.

1.2. Modèle d'accès

1.2.1. Les types d'information et les relations d'accès

Dans le modèle d'accès :

- les informations de la base de données sont regroupées en classes appelées OBJETS, une information d'une certaine classe est appelée REALISATION d'un objet.
- certains objets sont dits ELEMENTAIRES si leurs réalisations sont des valeurs; d'autres sont dits COMPLEXES si leurs réalisations ne sont pas des valeurs.
- Il est possible de décrire un type d'article ou d'enregistrement par un objet complexe le même, il est possible de décrire un champ (ou item ou zone ou attribut) par un objet élémentaire. Il existe toujours un objet particulier, la RACINE dont l'unique réalisation représente la totalité de la BD. Un objet élémentaire est dit COMPOSE si chaque réalisation peut être décomposée en éléments qui sont eux mêmes des réalisations d'objets élémentaires. Un objet élémentaire non décomposable est dit SIMPLE.
- Une RELATION d'une objet (appelé ORIGINE) vers un autre objet (appelé CIBLE) correspond à un chemin d'accès qui à partir de chaque réalisation origine permet d'accéder à un certain nombre de réalisations cibles. Une réalisation d'une relation est appelée une occurrence.

- une relation peut être définie entre deux objets complexes, d'un objet complexe vers un objet élémentaire, d'un objet élémentaire vers un objet complexe et de la racine vers un objet complexe.
- deux relations peuvent être déclarées inverses l'une de l'autre.
- La cardinalité d'une relation est caractérisée par le quadruplet I-J, K-L qui indique qu'à partir d'une réalisation origine (CIBLE), la relation permet d'accéder à un nombre de réalisations cibles (origine) compris entre I (K) et J(L).

Les différentes valeurs de I, J, K, L représentent des propriétés entre origine et cible, que nous retenons ici :

1) relations de la racine vers un objet complexe (OC)
C'est une représentation de l'accès séquentiel à tous les articles d'un même type dans une Base de données (BD), cette relation est caractérisée par le quadruplet 0-∞, 1-1.

2) Relations entre objets complexes (OC)
Ces relations représentent les types d'accès qu'on peut établir entre deux types d'articles, parmi lesquels il y a la relation "one to many", forte (0-∞, 1-1) ou faible (0-∞, 0-1).

3) Relations d'un objet complexe (OC) vers un objet élémentaire (OE)
Un objet élémentaire appartient toujours à un et un seul type d'articles, on peut toujours accéder aux valeurs de ses objets élémentaires associés. Ces valeurs peuvent être des valeurs simples ou des valeurs répétitives. Relations entre un OC et un OE identifiant est exprimé par le quadruplet 1-1, 0-1.

4) Relations d'un objet élémentaire vers un objet complexe
Un OE qui a cette propriété est une clé d'accès unique ou multiple. Une telle relation est une inverse de la relation OC vers OE et est caractérisée par les quadruplets :
   - clé d'accès unique : 0-1, 1-1
   - clé d'accès multiple : 0-∞, 1-1.

- Dans une relation entre objets complexes, si J > 1, on peut définir sur l'ensemble de cibles accessibles d'une même réalisation origine divers ordres d'accès : chronologique, antichronologique, tri par valeurs croissantes ou décroissantes d'objets élémentaires reliés à l'objet complexe cible par relation de quadruplet 1-1, 0-L, ...

1.2.2. Les primitives

1) La primitive d'accès :

La primitive d'accès permet à partir d'une réalisation O de son origine, d'accéder à un ensemble des réalisation cibles des occurrences de la relation ayant O pour origine. (Cette primitive réalise des accès répétitifs aux réalisations cibles d'un même type).

2) Les primitives de mise à jour

2.1. Primitive de création d'une réalisation d'objet complexe

Cette primitive crée une nouvelle réalisation d'un objet complexe et implique les vérifications de la cohérence de BD et éventuellement l'établissement des liens d'accès de manière à assurer le respect des propriétés de cardinalités.

2.2. Primitive de suppression d'une réalisation d'un objet complexe

Elle supprime toutes les occurrences de relation liées à cette réalisaton et ensuite elle supprime cette réalisation; elle réalise éventuellement d'autres modifications de manière à conserver la cohérence de la BD du point de vue de la cardinalité des relations.

2.3. Primitives de mise à jour associées aux relations d'accès

Dans le modèle d'accès, ces primitives permettent de définir des actions telles que :

- modifier le contenu d'un article
- créer (supprimer) un lien d'accès entre deux articles
- transférer un lien d'accès.

On a trois primitives :
- primitive de création d'une occurrence de relation
- primitive de suppression d'une occurrence de relation
- primitive de modification d'une occurrence de relation. Cette primitive s'applique uniquement aux relations telles que $I \neq 0$ ou $K \neq 0$.

Ces trois primitives s'appliquent aussi bien aux relations (OC, OE) qu'à celles de (OC, OC).

1.3. Le DML-SPHINX

Le DML-SPHINX a été constuit et défini selon les concepts du modèle d'accès. Il s'agit d'un langage d'exploitation de bases de données qui est associé au langage base COBOL.

C'est un langage de Boucle d'accès qui permet la sélection d'un ensemble de réalisations d'un objet complexe suivant les valeurs des objets élémentaires associés ou/et les autres relations d'accès entre objets.

° ° °

CHAPITRE II : CODASYL

2.1. Introduction

CODASYL a été défini par le Data Base Task Group (DBTG) qui a publié plusieurs rapports; dans notre étude nous nous basons essentiellement sur le rapport CODASYL-71.

Le DBTG a défini un DDL et un DML permettant de contrôler des notions relatives aux structures des données, types de données, contrôle d'autorisation d'accès, intégrité et performance d'une BD CODASYL. La structure des données CODASYL est celle du réseau, par opposition à la structure hiérarchique d'IMS.
COBOL a été choisi comme langage d'accueil du DML.

Dans ce chapitre, nous présentons les notions fondamentales CODASYL, les 2 notions "LOCK" et "Data-Base-Procedure" sont présentées séparément pour la clarté de l'exposé.

Il n'est pas possible, dans le cadre de ce mémoire, de donner un exposé complet des définitions CODASYL. Nous nous contenterons donc d'une présentation sommaire, en renvoyant le lecteur au rapport CODASYL-71.

2.2. Les notions fondamentales

2.2.1. Schéma

Un schéma est la description complète d'une base de données. Il se compose de la description des aréas, des types de record, des items associés et des types de set.

On désigne un schéma par un nom, ce nom est unique dans le SGBD.

2.2.2. Aréa

Une AREA est une subdivision nommée de l'espace mémoire de masse adressable dans la base de données et peut contenir des réalisations de types de records et des sets.

Une AREA peut être déclarée "provisoire" (AREA TEMPORARY)

2.2.3. Type de record et items associés

Un type de RECORD est une collection nommée de 0, 1 ou plusieurs items.

Il y a deux classes d'itmes : item composé et item élémentaire. Un item composé (DATA-AGGREGATE) est une collection nommée d'items associés à un type de record. Il y a deux types : vecteur et groupe répétitif. Un item élémentaire (par abréviation : item) (DATA-ITEM) est le plus petit élément nommé de type de données (c'est un élément atomique).

Une réalisation d'un type de record (par abréviation un record) est une unité d'accès logique de la SGBD-CODASYL.

Pour accéder à un record d'une BD, le SGBD doit avoir la possibilité de distinguer chaque record parmi les autres se trouvant dans cette BD. Cette distinction se fait par une DATA BASE-KEY associé à un record. La Data base-key est une espèce de référence (adresse) de record, elle est générée par le SGBD une fois un nouveau record est enregistré dans BD, cette valeur est unique et permanente (jusqu'à la disparition de ce record.)

Propriétés

A.- de type de record :

1- Localisation (WITHIN-clause).

Les réalisations d'un type de record peuvent être réparties en différents endroits (AREAS) d'une base de données. Ces endroits (AREAS) forment une zone où se trouvent des records de ce type.
S'il y a plusieurs aréas auxquels peuvent se trouver les réalisations d'un type de record, on indique, dans un programme DML, l'aréa voulue par un identificateur de l'aréa (AREA-ID).

2- Mode d'emplacement (LOCATION MODE)

Un nouveau record peut être rangé suivant un des trois modes :

- direct : laisser au SGBD le soin de trouver une place libre ou indiquer par référence (DB-KEY) la place voulue au SGBD
- CALC : ranger par clé (valeur d'un (ou plusieurs) item(s) forme(nt) une clé par concaténation) unique ou multiple (CALC DUPLICATES NOT ALLOWED, CALC DUPLICATES ALLOWED)
- VIA : placer tout près d'une réalisation "OWNER" (Voir 2.2.4.)

LOCATION MODE détermine la position d'un record dans une BD (relative à une aréa), ce qui permet d'utiliser un certain nombre de moyen d'accès.

B.- des items associés

1- Les classes de valeurs des items (élémentaires)

Les valeurs d'un item peuvent être des valeurs numériques (BINARY/DECIMAL, FIXED/FLOAT, REAL/COMPLEX), alphanumérique (PICTURE), bit ou caractère (BIT/CHARACTER), DATABASE-KEY, ou un type défini par l'implémentateur.

2- RESULT

Les valeurs d'un item (élémentaire) peuvent être le résultat de l'exécution d'une procédure qui peut utiliser éventuellement :

- les valeurs autres items, ces autres items peuvent se trouver dans ce même type de record ou/et dans les "members" du type de set dont l'"owner" est le type de record auquel appartient cet item.
- ou bien un ou plusieurs type member du type de set dont l'owner est le type de record auquel appartient cet item.

Le résultat peut être enregistré physiquement (ACTUAL) ou non (VIRTUAL) dans la base de données. Si c'est ACTUAL, ce résultat peut être modifié par ordre MODIFY/STORE/DELETE/INSERT/REMOVE. Si c'est VIRTUAL, chaque fois on fait une lecture (GET), le SGBD va fournir un résultat effectuant la procédure spécifiée.
Les valeurs de cet item sont le résultat d'une procédure, effectué par le SGBD, en conséquence, l'utilisateur ne peut pas modifier ses valeurs.

3- SOURCE

La valeur de cet item est celle d'un item "source" appartenant à l'"owner" d'un type de set-spécifié dans la clause OF OWNER OF - dont un "member" est le type de record auquel appartient cet item.

Les valeurs de cet item peuvent être enregistrées (ACTUAL) ou non (VIRTUAL) dans la base de données.

Si un record auquel appartient cet item ne participe pas à l'ensemble des occurrences du type de set relatif à l'option SOURCE, sa valeur est nulle.

Si l'origine (SOURCE) n'est pas un item de contrôle (implicitement ou explicitement déclaré dans la clause SET OCCURRENCE SELECTION de la description de cetype de set), on ne peut pas modifier directement le contenu de cet item.

Par contre, si l'origine (SOURCE) est un item de contrôle, on peut modifier les valeurs de cet item mais cela implique une modification des occurrences du type de set spécifié.

4- CHECK

Pour vérifier la validité des valeurs d'un item (contrainte d'intégrité des valeurs d'un item) ou pour annuler la conversion entre la représentation des valeurs d'un item du schéma et celle des valeurs de ce même item utilisé dans un sub-schéma (l'option "PICTURE").

On peut demander un ou plusieurs contrôle(s) de validation des valeurs d'un item :

- les valeurs d'un item doivent se trouver entre les deux limites (inférieure et supérieure) (RANGE)
- par une procédure utilisant une variable pour indiquer le résultat de contrôle et éventuellement un ou plusieurs items appartenant au même type de record.

Si cet item est déclaré VIRTUAL RESULT, le contrôle de validité se passe au moment de lecture (GET).

Si ce item est déclaré ACTUAL RESULT, le contrôle se passe quand la valeur de cet item, est modifiée; implicitement ou explicitement.

Si l'on utilise les deux contrôles, il existe un ordre de contrôle (RANGE est le premier).

5- ENCODING/DECODING

On peut demander une procédure pour transformer (encoder/décoder) la représentation des valeurs d'un item se trouvant dans la BD (représentation interne) par rapport à la représentation externe de cet item (dans un subschéma). Cette procédure va faire une conversion chaque fois que l'on demande un accès ou une mise à jour.

6- OCCURS

Un item composé peut être un vecteur ou un groupe répétitif.

7- Argument d'un critère d'accès.

Les valeurs d'un (plusieurs) item(s) servent comme clé d'accès aux records d'un certain type (LOCATION MODE IS CALC) ou aux records member (SERACH-KEY ou Set indexé, voir 2.2.4.)

8- Argument d'un critère de sélection

Les valeurs des items peuvent être des arguments d'un critère de sélection d'un set. (ces items sont cités dans la clause SOS, voir la suite).

2.2.4. Type de set

Un type de set est une collection nommée des types de record Chaque type de set a un et un seul type OWNER et un ou plusieurs types MEMBER.

Un type de set "dynamique" est un type de set dans lequel tous les types de record du schéma, sauf son owner, peuvent potentiellement être ses members.

Un type de set singulier est un type de set dont l'owner est SYSTEM. Ce type de set a un seul set.

Un set est un ensemble des records dont un record est celui de type owner et 0, 1 ou plusieurs record member.

Propriétés

1. Un record member d'un type de set ne peut appartenir qu'à un set de ce type.
2. Un type de set CODASYL correspond à une relation "one to many", forte (MANDATORY AUTOMATIC) ou faible.
   (relation "one to one" est un cas particulier de "one to many").
3. Mode d'organisation d'un set :
   il y a plusieurs modes d'organisation : par liste (et on peut demander une chaine inverse - MODE IS CHAIN (LINKED TO PRIOR)), par vecteur de pointeur (POINTER-ARRAY), par vecteur de pointer 'dynamique' (POINTER-

ARRAY DYNAMIC) ou par un mode défini par l'implémenteur.
Un type de set "dynamique" est celui déclaré avec "MODE IS POINTER-ARRAY DYNAMIC".

4- Relation inverse
A l'exception de set "dynamique", il existe toujours une relation inverse.
Les modes d'organisation des sets CODASYL (Le "COSET") permettent de déduire cette propriété. En effet, sauf le mode POINTER-ARRAY DYNAMIC, dans les autres modes, il y a toujours au moins un chemin d'un member vers son owner.

5- L'ordre d'un set : on peut définir un ordre dans un set : chronologique (ORDER IS LAST), antichronologique (ORDER IS FIRST), avant ou après le "courant" (x) du set (ORDER IS PRIOR/NEXT) ou un ordre de tri :

- Selon les valeurs croissantes de database-key des records member appartenant au type de set spécifié;
- S'il y a plusieurs types members, les réalisations de chaque type member, appartenant au type de set spécifié, peuvent être triées séparément des autres (clause WITHIN RECORD-NAME), soient par les valeurs croissantes de database-key, soient par les valeurs croissantes/décroissantes des items de chaque type member (spécifiés dans la clause ASCENDING/DESCENDING KEY)
- Selon les valeurs croissantes/décroissantes des items appartenant au type member. Dans ce cas :
  Il est possible de déterminer un "record index" qui permet d'indexer les records member par clé de tri. (clause INDEXED NAME is index-name).
  On peut réaliser un index à plusieurs niveaux (RANGE KEY). Cette RANGE KEY peut fournir un critère de sélection de set (pour SOS)
- Une clé de tri peut être unique (DUPLICATES NOT ALLOWED) ou multiple (DUPLICATES ALLOWED); quand c'est une clé multiple, on peut spécifier l'ordre d'insertion dans le set spécifié (FIRST/LAST).

(x) Durant l'exécution d'un programme DML, chaque type de set a un "courant de set", il permet de positionner le set courant (désigné par la Database-Key d'un record appartenant à ce set) de ce type.

6- Insertion automatique d'un record member dans un set (MEMBER IS AUTOMATIC) chaque fois qu'un record member est créé, il est inséré automatiquement dans tous les sets dont le type de ce record est member automatique. Inversément, on peut déclarer un MEMBER MANUAL. (La création d'un record n'implique pas une insertion dans un set de ce type, elle doit être réalisée explicitement par l'ordre INSERT).

7- MEMBER IS MANDATORY : on ne peut pas utiliser l'ordre de suppression de lien REMOVE. En plus un record member doit appartenir à un set de ce type durant toute sa vie.
Inversément, l'option MEMBER IS OPTIONAL permet d'utiliser cet ordre.
Dans tous les cas, la suppression des liens est automatique quand une réalisation owner ou member disparaît.

8- Critère d'accès aux réalisations member : D'une part, on peut définir un critère d'accès aux réalisations member (SEARCH KEY). Soit par clé calculé (l'option CALC) soit par clé indexée soit par une procédure. D'autre part, on peut déclarer un index d'une clé de tri (set indexé). (Ces critères peuvent intervenir dans l'ordre FIND (Format 6). S'il existe une SEARCH KEY ou un index de set, la recherche est effectuée suivant le critère déterminé, sinon c'est un balayage séquentiel).

9- DUPLICATES NOT ALLOWED : Dans un set, les valeurs de certains items d'un type member peuvent être uniques pour les records member (DUPLICATES NOT ALLOWED) ou non.
Cette clause permet de déclarer l'unicité des valeurs d'items, relatif à un set de ce type. Les items peuvent être utilisés dans le SOS.

10- L'établissement d'un chemin d'accès et l'éventuelle détermination des paramètres d'accès.
(SET OCCURRENCE SELECTION-SOS-) : il y a deux modes de sélection :
- soit par le courant du set
- soit par un chemin d'accès (déterminé implicitement (Format 1) ou explicitement (Format 2)).

Dans ce mode, on indique (implicitement ou explicitement par la clause USING ou ALIAS) les arguments des critères de sélection.

Dans le cas où il y a une ambiguité dans l'indication des arguments on emploie la clause ALIAS
La SOS sert à sélectionner un set, soit pour insérer un record member dans ce set, soit pour accéder à un record member; cette sélection est réalisée par le système.

2.2.5. Sub-schéma

Un SUB-Schéma est une description partielle d'une base de données, basée sur la description du schéma auquel il appartient. Il peut y avoir une modification de certaines propriétés des aréas, des types de record, des items associés et des types de set du schéma, appartenant à ce subschéma. On désigne un subschéma par un nom.

Propriétés

1- Synonymes : on peut désigner un même type de données qui se trouve dans un schéma par un autre nom dans un sub-schéma du schéma spécifié. (RENAMING SECTION).
2- AREA SECTION : On détermine les AREAS utilisés par un SUB-SCHEMA ainsi que leurs propriétés ("LOCK")
3- RECORD SECTION : On y énumère et définit les types de record et les items associés, du schéma, utilisés dans un subschéma.
   a) énumération des types de record du subschéma : on les énumère par un nom associé à un nombre de niveau 01 (pour distinguer avec ses subordonnés)
   b) détermination des aréas dans lesquelles se trouvent des réalisations d'un type de record du subschéma
   c) détermination des items associés à un type de record du subschéma.
      - on les désigne par un nom associé à un nombre de niveau
      - on peut changer l'ordre de désigantion des items ou items composés.
      - au niveau d'item composé :
        on peut redéfinir un vecteur comme un tableau à plusieurs dimensions (3 au maximum), de longueur variable, avec une (plusieurs) indice(s) ou non.
        On peut former un nouveau item composé en regroupant des items ou items composés du schéma correspondant.
        On désigne ce nouvel item par un nouveau nom.
      - au niveau d'item (élémentaire): on peut changer certaines propriétés USAGE : on définit le format de types de données COBOL (COMP-n, COMP, DISPLAY, DATABASE-KEY) apparu dans la UWA. (C'est comme dans la clause USAGE DE COBOL). Ce format peut être différent du format défini dans le schéma.

PICTURE : On définit les caractéristiques générales de types de données COBOL apparu dans VWA (c'est comme la clause PICTURE de COBOL). Ce format peut être différent du format défini dans le schéma.

SIGN LEADING/TRAILING (SEPARATE) : on spécifie la position et le mode de représentation du signe +/-. Ce signe peut être inclu ou exclu dans la représentation des valeurs d'un item numérique.

VALUE : on peut spécifier un nom-condition avec des valeurs associées qui peuvent définir une (plusieurs) borne(s).

4- SET SECTION : pour énumérer et définir les types de set du schéma utilisés dans un subschéma.
On peut redéfinir un (plusieurs) autre(s) critère(s) de sélection de set.

2.2.6. Contrôle d'autorisation

L'administrateur peut demander un contrôle d'autorisation associé à un ordre du DML ou celui effectué sur un schéma ou un subschéma. Nous établissons la table suivante :

NIVEAU	OPERATION	MODE D'USAGE	SIGNIFICATION	MOYEN DE CONTROLE
SCHEMA	LOCKS DISPLAY COPY ALTER		- Accéder/créer/changer les moyens de controle - Accéder au schéma, sauf les moyens controle - copier des descriptions dans un subschéma - changer des descriptions du schéma, sauf les moyens de contrôle	- Verrous - zone contenant des verrous - procédure
AREA	(open)	RETRIEVAL EXCLU.RETRIEV. PROTECTED " UPDATE EXCLU. UPDATE PROTECTED " Support-function	- mode de lecture avec partage - de même, mais sans partage - refus des mises à jour des autres users - mise à jour - mise à jour, sans partage - mise à jour, accepter le partage en mode RETRIEVAL - défini par l'implémenteur	
Type de record	A [INSERT REMOVE STORE DELETE DELETE DELETE DELETE MODIFY FIND GET]	 ONLY SELECTIVE ALL	- insérer dans un set - suppression d'un lien d'un set - création d'un record - suppression d'un record, s'il n'y a pas de record member - suppression owner avec member mandatory - suppression avec sélection member optional - suppression owner et tous member - modification un record tous member - accéder à un record - lecture des valeurs des items	
Items associés	B [STORE GET MODIFY]		- même signification, mais au niveau des items associés	

NIVEAU	OPERATION	MODE D'USAGE	SIGNIFICATION	MOYEN DE CONTROLE
Type de set	C [ORDER INSERT REMOVE FIND]		- changer l'ordre d'un set - insérer un record dans un set - retirer un record d'un set - accéder à un record via ce set	
SUB-SCHEMA	D [LOCKS DISPLAY ALTER COMPILE]		] - même signification que dans le schéma - compiler un programme utilisant ce subschéma	

TABLE 2.1.

2.2.7. Traitement associé à un ordre CODASYL

Avant ou après chaque ordre CODASYL sur les aréas, des types de record, des items associés, des types de set d'un schéma...on peut demander l'exécution d'une procédure.
Remarquons qu'une seule procédure peut être associée à un ou plusieurs ordre (s) DML-CODASYL (à l'exception des procédures de contrôle d'autorisation).
Ce sont des procédures pour :

- effectuer des contrôles d'autorisation
- calculer des data-base-key
- calculer des valeurs des items
- faire une recherche d'un record
- etc ...

On peut diviser les procédures en 3 classes :

1) les procédures de gestion standard d'un SGBD, ces procédures sont connues par le SGBD
2) les procédures liées à des traitements particuliers : RESULT, CHECK, ENCODING, DECODING, ... ont pour but d'économiser la programmation, de contrôler des contraintes d'intégrité ...
3) D'autre part, l'administrateur peut spécifier des procédures non standards.

Nous présentons sous forme de table les procédures associées aux ordres CODASYL. Pour la clarté de la présentation, nous établissons 2 tables séparées, une pour la clause ON, et une pour les autres procédures.

	OPERATION	MODE D'USAGE	SIGNIFICATION
AREA	OPEN	EXCLUSIVE	- utilisation d'aréa sans partage
		PROTECTED	- utilisation d'aréa avec protection de mise à jour
		NON-EXCLUSIVE	- utilisation d'aréa avec partage
		UPDATE	- avec mise à jour
		EXCLUSIVE UPDATE	
		PROTECTED UPDATE	
		NON EXCLUSIVE UPDATE	- voir aussi table 2.1.
		RETRIEVAL	
		EXCLUSIVE RETRIEVAL	
		PROTECTED RETRIEVAL	
		NON EXCLUSIVE RETRIEVAL	
	CLOSE		
Type de record	INSERT		
	REMOVE		
	STORE		
	DELETE		- voir la table 2.1.
	DELETE	ONLY	
	DELETE	SELECTIVE	
	DELETE	ALL	
	MODIFY		
	FIND		
	GET		
Item ou groupe d'item	STORE		
	GET		- voir la table 2.1.
	MODIFY		
Type de Set	ORDER		
	INSERT		- voir la table 2.1.
	REMOVE		

TABLE 2.2. : CLAUSE "ON"

	OPERATION	USAGE	SIGNIFICATION
Type de record	STORE/FIND voir table 2.1(A)	LOCATION MODE CALC contrôle d'autorisation	calculer l'adresse basée sur les valeurs des items du record
Items ou groupes d'items	MODIFY,STORE,DELETE INSERT,REMOVE GET STORE,GET MODIFY,STORE/FIND voir table 2.1(B)	ACTUAL RESULT VIRTUAL RESULT CHECK ENCODING/DECODING contrôle d'autorisation	calculer valeurs d'item contrôler la validité des valeurs d'item transformer la représentation des valeurs d'item
Type de set	STORE/MODIFY/FIND voir table 2.1(C)	SEARCH KEY contrôle d'autorisation	Calculer la clé d'accès de record member
SUB-SCHEMA	voir table 2.1(D)	Contrôle d'autorisation	

TABLE 2.3.

2.2.8. Variable de travail

CODASYL utilisant des variables de travail qui n'appartiennent pas à un type de record comme AREA-ID, ou les variables dans les clauses LOCATION MODE DIRECT, ALIAS; mais elles sont liées à ce type de record pour effectuer des mécanismes de gestion.

D'autre part, il y a des variables liées aux procédures pour enregistrer des codes d'erreur de retour.

2.3. *Les relations d'accès*

2.3.1. Relations d'une AREA vers un type de record

A partir d'une aréa spcéifiée, on peut accéder aux records d'un type qui appartiennent à cette aréa.

2.3.2. Relations entre types de record

A partir d'un record owner, on peut accéder aux records member via un set. De même, à partir d'un record member, on peut accéder à un record owner via un set auquel ils appartiennent (à l'exception d'un set "dynamique")

2.3.3. Relation d'un type de record vers des items

Les items sont des constituants d'un type de record. On peut toujours accéder aux valeurs des items du record.

2.3.4. Relation d'un item ou plusieurs items vers un type de record

C'est un accès par clé unique ou multiple. Il y a 2 cas :

- clé d'accès à un type de record (CALC) : on peut accéder directement au(x) record(s) d'un certain type en fournissant des valeurs des items (constituant cette clé) aux SGBD.
- accès par clé via un set : à partir d'un record owner, on peut accéder à un (ou plusieurs) record(s) member par une clé d'accès de set.

D'autre part, on peut accéder à un record par une référence (DATABASE-KEY).

° ° °

CHAPITRE III : UN MODELE CODASYL SIMPLIFIE EN VUE DE LA CONSTRUCTION UN DML DE TYPE DML-SPHINX

3.1. Introduction

Nous allons construire un modèle CODASYL simplifié à partir du :
- modèle d'accès et DML-SPHINX, en tenant compte des facteurs suivants :
  - le modèle d'accès est un modèle relationnel binaire;
  - il existe dans ce modèle des notions comme la clé d'accès, l'identifiant, la cardinalité des relations d'accès ...
  - la structure et la sémantique du DML-SPHINX.
- modèle CODASYL, (avec DDL et DML) en tenant compte des caractéristiques suivantes :
  - certaines propriétés sont destinées au DB administrateur seulement (les paramètres physiques, les "Data-Base-procédure" ...)
  - une même notion est répartie en différentes clauses du DDL
  - certaines propriétés sont associées au DML-CODASYL.

Notons que DML-CODASYL est un langage "ponctuel" dans le sens que chaque ordre DML s'effectue sur un record d'un type donné tandis que DML-SPHINX est un langage de haut niveau.

3.2. Les types d'information et les relations d'accès

3.2.1. Base de données (BD)

Une base de données est le contenant de toutes les réalisations de types de record, des occurrences des types de set et des aréas contrôlés par un schéma spécifique. Chaque base de données a son propre schéma. L'utilisation d'une base de données est partageable.

3.2.2. AREA

Une aréa est une subdivision de l'espace de mémoire d'une base de donnée; elle peut contenir des réalisations des types de record et des occurrences des types de set.

L'utilisation d'une aréa est partageable entre différents utilisateurs. Il y a 6 modes d'usage d'un aréa :

RETRIEVAL : consultation avec partage entre plusieurs utilisateurs, y compris la mise à jour (en mode PROTECTED UPDATE ou UPDATE) des autres utilisateurs;

UPDATE : mise à jour avec partage, même en mode UPDATE

PROTECTED RETRIEVAL : consultation et protection contre une mise à jour

EXCLUSIVE RETRIEVAL : consultation sans partage

PROTECTED UPDATE : mise à jour avec partage en mode RETRIEVAL

EXCLUSIVE UPDATE : mise à jour sans partage.

Correspondance et justification

Aréa provisoire (AREA IS TEMPORARY).

Un espace mémoire de masse est créé au début de RUN-UNIT et disparu à la fin de RUN-UNIT. Les changements ne sont pas enregistrés dans la base de données.

Les aréas "provisoires" peuvent être utiles dans deux cas suivants :

1) la mise au point d'un programme

2) ces aréas sont utilisées uniquement en mode de consultation.

Mais l'utilisation d'une base de données, en particulier celle d'un aréa, "provisoire" ou permanent, est sous la responsabilité d'un DBA, en conséquence nous n'indiquons pas cette propriété à l'utilisateur.

3.2.3. Les types de record et items associés

A.- Type de record

Un type de record est un ensemble constitué d'item fictif "aréa" (voir 3.2.6.) et de 0, 1 ou plusieurs items ou items composés pour définir un même type d'entité, un même type d'individu.

Un record est une réalisation d'un type de record contenant les valeurs de ses items. Un record est unité d'accès logique du SGBD.

Les réalisations d'un type de record peuvent être réparties dans une ou plusieurs aréas.

Deux records sont discernables, même s'ils sont de même type et ont

les mêmes valeurs d'items associés. (voir correspondance et justification (1))

On ne peut pas modifier des réalisations d'un certain type de record. (2)

Un type de record peut avoir un (ou plusieurs) identifiant(s) (voir la suite de ce paragraphe).

Un type de record peut avoir une (ou plusieurs) clé(s) d'accès (voir la suite de ce paragraphe).

Correspondance et justification

1) C'est une propriété du SGBD-CODASYL, en effet un record est une unité d'enregistrement, les manipulations des valeurs d'items associés s'effectuent sur des "champs" correspondants de ce record.
2) Pour assurer que l'utilisateur DML-MCS écrit correctement les ordres de mise à jour des valeurs d'items ou des records d'un certain type, nous devons signaler aux utilisateurs les éventuelles restrictions des ordres DML-MCS relatifs à un type de record ou à ses items ou à un type de set auquel appartient ce type de record.

En effet, les items déclarés dans un schéma (Codasyl) n'appartiennent pas nécessairement à un subschéma de ce schéma. Cela peut avoir les conséquences suivantes :

- la génération des ordres DML-CODASYL (effectuée par le compilateur DML-MCS) peut impliquer une utilisation d'un chemin d'accès, ce dernier a besoin d'un des items qui n'appartient pas à un subschéma spécifié.
- un item participant à une clé de tri d'un set n'appartient pas à un subschéma spécifié
- un item déclaré dans la clause LOCATION MODE IS CALC ... n'appartient pas à un subschéma spécifié.

En général quand un item utilisé dans un mécanisme de gestion qui implique un certain contrôle apparait dans une des clauses : SEARCH-KEY, DUPLICATES NOT ALLOWED, clé de tri ...) n'appartient pas à un subschéma, il peut avoir des conséquences (des erreurs d'exécution éventuellement) ces dernières dépendent des particularités du SGBD réalisant les propositions Codasyl. Ce SGBD peut imposer des restrictions d'emploi des ordres DML-CODASYL, relatifs à un type de record ou à un item associé ou à un type de set auquel appartient ce type de record. Ce qui entraine des res-

trictions des ordres DML-MCS de même classe.

Pour simplifier nous nous contentons de siganler la non-modification des réalisations d'un type de record chaque fois une des conditions d'exécution correcte des ordres de mise à jour (relatifs à un type de record, un item associé ou un type de set auquel appartient ce type de record) ne sont pas assurée.

3) La simplification de la clause LOCATION MODE.
Le LOCATION MODE a deux propriétés : le rangement d'un nouveau record et l'éventuelle participation à un chemin d'accès. L'étude de SOS nous dit qu'on peut négliger la clause SOS (voir 3.2.4.), ici nous n'examinons que la propriété de rangement.

LOCATION MODE DIRECT : dans CODASYL, on peut utiliser soit une variable soit un item, de type DATABASE-KEY, qui contient une valeur initialisée avant la création d'un record. Lors de la création si cette valeur est nulle, le SGBD génère une valeur lui-même, sinon il va essayer d'enregistrer ce record dans la BD selon la valeur indiquée (normalement c'est un numéro de page). Ainsi CODASYL offre à l'utilisateur la possibilité de gérer lui-même le placement d'un nouveau record dans la BD.

Dans le cas d'une variable déclarée dans LOCATION MODE DIRECT, nous proposons de ne pas informer de cette possibilité les utilisateurs DML-MCS et nous laissons au compilateur DML-MCS le soin de gérer cette variable.

Dans le cas d'un item du type DATABASE-KEY, nous devons indiquer la signification de cet item par rapport au type de record vers lequel il va pointer.

LOCATION MODE CALC : nous le transformons en une clé d'accès et éventuellement en un identifiant (voir 3.2.3.C. et 3.2.3.D.). Dans le cas où un de ces items n'appartient pas à un schéma, nous devons signaler aux utilisateurs.

LOCATION MODE VIA : le rangement s'effectue le plus près possible du point logique d'insertion; c'est une caractéristique interne de la définition d'un schéma CODASYL (pour une certaine performance); pour des raisons de simplicité, les programmeurs n'auront pas à la connaître.

Ainsi nous proposons de négliger LOCATION MODE.

4) La transformation de la clause WITHIN ... (AREA-ID) : nous le transformons en un item fictif associé à chaque type de record (voir la suite de ce paragraphe).

B.- Items associés

Il y a 2 classes d'items associés à un type de record : item composé et item élémentaire. Un item composé est un ensemble constitué par un ou plusieurs items composés ou items élémentaires.
Il y a 2 types d'items composé : vecteur et groupe répétitif.
- un vecteur est un item dont plusieurs valeurs sont associés à un record
- un groupe répétitif est un ensemble constitué par les items élémentaires ou/et des items composés à un record dont plusieurs valeurs sont associées à un record.

Un item élémentaire (par abréviation : item) est le plus petit élément de type de données.
Les réalisations d'un item sont des valeurs.
Les réalisations d'un item composé sont des valeurs de ses constituants.

Le domaine de valeurs d'un item composé est défini par un ensemble ordonné de domaines des items élémentaires constituants.

La longueur d'une valeur d'un item composé peut être variable, en conséquence celle d'un record peut être variable. (voir "correspondance et justification (1))

Parmi les items associés à un type de record, il y a l'item "aréa" qui est un item fictif et a pour but d'identifier une aréa dans un ordre DML-MCS, chaque type de record a son item "aréa".(2)

Les valeurs d'un item peuvent être :

- soumises à un contrôle d'intégrité (3)
- le résultat d'une procédure basée sur des valeurs des autres items ou/et des records member (4)
- celles d'un autre item appartenant à un type owner (5)
- un ensemble d'items peut être un identifiant de type de record ou de set (voir la suite)
- un ensemble d'items peut être une clé d'accès, de type de record ou de set (voir la suite)

Correspondance et justification

1) CODASYL admet des items composés de longueur variable
2) Item "aréa" (transformation de la clause WITHIN)
   Pour nous confirmer avec les formes des ordres REACH et PREPARE du DML-MCS (basé sur le DML-SPHINX), nous transformons la clause WITHIN, avec ou sans AREA-ID, de CODASYL en l'item fictif "identifiant aréa" avec une liste des aréas spécifiés dans la clause WITHIN.

   En plus, l'item "aréa" a le même rôle de AREA-ID (CODASYL) dans le cas où il y a plusieurs aréas, ce qui rend inutile la connaissance de AREA-ID vis-à-vis de l'utilisateur.
3) Simplification de la clause CHECK
   La clause CHECK (CODASYL) permet d'annuler une conversion entre la représentation des valeurs d'un item dans la BD (SCHEMA) et celle à l'extérieur (SUB-SCHEMA) ou de vérifier la validité des valeurs d'un item.

   Une conversion ou non des différentes représentation des valeurs est une caractéristique interne d'un schéma. Nous ne retenons qu'ici le contrôle de validité des valeurs d'un item. Le contrôle peut être réalisé par une procédure.

   Un variable de travail associé à cette procédure pour contenir le code d'erreur de retour, pour simplifier, nous laissons le soit au compilateur DML-MCS de traduire ce code d'erreur et de prendre éventuellement des décisions
4) Simplification de la clause RESULT.
   Les valeurs d'un item peuvent être le résultat d'un calcul représenté par un nom de procédure. La notion de procédure du SGBD est nécessaire pour la compréhension du déroulement d'un programme, de la description d'un subschéma DDL-MCS; tandis que la connaissance de nom de procédure n'apporte rien

aux utilisateurs. En conséquence, nous proposons de conserver seulement la notion de procédure du SGBD.

Les valeurs de cet item peuvent être enregistrées ou non dans la BD (l'option ACTUEL/VIRTUAL), c'est une caractéristique physique du schéma correspondant, en conséquence, l'utilisateur n'a pas besoin de connaître.

5) Simplification de la clause SOURCE
De même, on peut simplifier l'option ACTUAL ou VIRTUAL de cette clause.

C.- Notion d'identifiant de type de record

L'identifiant de type de record est un ensemble d'items associés, tel que chaque valeur de cet ensemble est unique parmi ses valeurs enregistrées ans des records de ce type. Un type de record peut avoir plusieurs identifiants. C'est un identifiant global (par rapport à l'identifiant de set).

Correspondance et justification

Il y a 2 cas où un item ou un groupe d'items est transformé en un identifiant.

1. Dans un schéma CODASYL, un type de record est déclaré avec "LOCATION MODE IS CALC ... DUPLICATES NOT ALLOWED"
2. Quand un type de record est "MEMBER MANDATORY AUTOMATIC" d'un type de set singulier (c'est à dire "owner is system") et avec une ou plusieurs clauses DUPLICATES NOT ALLOWED.

En effet :
Dans le premier cas, les valeurs d'items sont déclarées unique (DUPLICATES NOT ALLOWED) pour chaque réalisation du type de record auquel appartiennent ces items, par rapport à un aréa spécifié (ce qui entraîne que s'il y a plusieurs aréas déclarés dans la clause WITHIN, l'identifiant inclut l'item fictif "aréa").
Dans le deuxième cas, normalement ces items sont des identifiants de set (voir la suite de ce chapitre) mais :
- un type de set singulier contient un unique set, avec l'owner SYSTEM
- une réalisation d'un type member MANDATORY AUTOMATIC appartient à ce set depuis sa création jusqu'à sa disparition.

En conséquence, ces items sont un identifiant de ce type de record.

Cet identifiant se compose uniquement de ces items, même dans le cas où il y a plusieurs aréas déclarées dans la clause WITHIN du type member car il s'agit ici d'un type de set.

Puisqu'il existe éventuellement plusieurs ensembles d'items déclarés avec la clause DUPLICATES NOT ALLOWED, donc un type de record peut avoir plusieurs IDENTIFIER.

En plus, pour que l'utilisateur puisse écrire correctement ses programmes (ne pas faire créer deux records avec une même valeur de l'identifiant) il fallait l'informer de l'existence des items déclarés dans les clauses DUPLICATES NOT ALLOWED

D.- Notion de clé d'accès de type de record

Une clé d'accès est un ensemble d'items associés, tel qu'il existe un moyen rapide de retrouver, à partir d'une valeur, formé par les valeurs de ces items, tous les records qui les contiennent.

Un type de record peut avoir plusieurs clés d'accès.

Correspondance et justification

1) Dans le cas où on a "LOCATION MODE IS CALC ..." et plusieurs aréas dans la clause WITHIN, la clé d'accès est formée par l'item "aréa" et les items déclarés dans cette clause car il s'agit ici d'une clé "calculée" relative à un aréa; nous devons la transformer en un identifiant global.

   S'il n'y a qu'une seule aréa dans la clause WITHIN, la clé est formée par les items déclarés dans cette clause uniquement.

2) Dans le cas où on a :
   - un set singulier
   - le type member est déclaré MANDATORY AUTOMATIC et une des clauses suivante :
     . ORDER IS SORTED INDEXED ... $\left\{\begin{matrix}A\\D\end{matrix}\right\}$ KEY IS
     . SEARCH KEY IS ...

   On obtient donc une (ou plusieurs) clé d'accès (il peut exister plusieurs cas correspondant aux caractéristiques citées ci-dessus)

3) Les spécifications des mécanismes d'accès ou d'insertion (par index, par

algorithme) sont les caractéristiques internes d'un SGBD. L'utilisateur n'a pas besoin de connaître ces mécanismes.

4) Notons qu'un identifiant de type de record n'est pas nécessairement une clé d'accès (l'option DUPLICATES NOT ALLOWED permet de déclarer seulement l'unicité des valeurs d'items associés)

3.2.4. Les types de set

A un type de set correspond une relation d'accès associée à un couple de types de record appelés type owner et type member de la relation (1)

Un type owner est un type de record origine d'un type de set, de même un type member est un type de record cible. Le type owner d'un type de set ne peut être le type member du même type de set.

Un set est un ensemble non vide de records dont un record est un record owner et les autres sont des records members.

Un record owner est une réalisation du type owner.

Un record member est une réalisation du type member, appartenant à un set de ce type.

Un set vide est un set qui n'a pas de record member.

Un record member d'un type de set appartient à un seul set de ce type.

Un type de set singulier est un type de set dont type owner est SYSTEM (c'est à dire SGBD). Ce type de set a un seul set.

- Un type de set "dynamique" est un type de set dont le type member n'est pas défini, les records de différents types peuvent devenir member de ce type de set (sauf ceux de son owner).

Un type de set est une relation 1-n ("one to many") forte (MANDATORY AUTOMATIC) ou faible.

D'autre part, un type member peut avoir des propriétés suivantes :

- MANDATORY : on ne peut pas utiliser l'ordre DETACH pour supprimer le lien d'accès d'un record member. En plus un record member doit appartenir à un set de ce type durant toute sa vie.
- OPTIONAL : inversément, on peut utiliser cet ordre
- AUTOMATIC : insertion automatique de lien d'accès lors de la création d'un record member
- MANUAL : inversément, on doit insérer d'un lien d'accès par l'ordre ATTACH

A l'exception d'un type de set dynamique, il existe toujours une relation inverse entre type OWNER et type member d'un type de set.

L'ordre d'un set : il y a 3 ordres possibles :

- ordre chronologique
- ordre antichronologique
- ordre de tri.

Un tri peut s'effectuer selon un des 3 modes :

- tri par référence (DATA BASE KEY)
- tri par ordre croissant de valeurs d'un groupe d'items
- tri par ordre décroissant de valeurs d'un groupe d'items.

Correspondance et justifications

1) Dans CODASYL, un type de set a un type owner et un ou plusieurs types member (sauf type de set dynamic qui n'a pas de type member explicite). Dans le modèle d'accès, une relation d'accès est une relation binaire, d'autre part, l'ordre d'accès et les ordres de mise à jour du DML-SPHINX sont basés sur des relations binaires. Ce qui justifie la transformation suivante :

   Si un type de set CODASYL a plusieurs types member, nous le transformons en plusieurs types de set qui ont un type owner et un type member.

2) Quand un type de set CODASYL est déclaré avec mode d'organisation POINTER-ARRAY DYNAMIC nous le transformons en un type de set "dynamique".

A partir des modes d'organisation (MODE IS CHAIN, MODE IS CHAIN LINKED TO PRIOR, MODE IS POINTER-ARRAY, MODE IS POINTER ARRAY DYNAMIC) nous déduisons les propriétés suivantes :

- un type de set "dynamique" n'a pas de relation inverse
- sauf le type de set "dynamique" tous les types de set CODASYL ont une

relation inverse, car d'une part dans les trois modes d'organisation (MODE IS CHAIN, MODE IS CHAIN LINKED TO PRIOR, MODE IS POINTER-ARRAY) pour chaque set il existe au moins un chemin d'accès vers son owner, d'autre part, l'ordre d'accès CODASYL (FIND) permet d'accéder à un record owner en se basant sur le "courant de set".

Mais les modes d'organisation en soi ne sont que des propriétés physiques ce qui entraînte le "camouflage" de ces propriétés vis-à-vis des utilisateurs.

3) Quand un type de set CODASYL est déclaré avec OWNER IS SYSTEM, il y a des cas où on peut camoufler ce type de set à condition qu'il ne possède pas de propriété utile aux utilisateurs. Cela veut dire que si dans ce type de set :

   - MEMBER est MANDATORY AUTOMATIC
   - il n'y a pas ordre logique (chronologique, antichronologique, tri)

   On peut le camoufler car :

   - il n'y a pas de type de record owner (type owner est généré et géré par SGBD)
   - ce type de set a un seul set et chaque record member appartient à ce set depuis sa création jusqu'à sa disparition (MANDATORY AUTOMATIC), l'utilisateur n'effectue pas les opérations d'insertion ou d'enlèvement des liens d'accès. C'est-à-dire il n'y a pas de manipulation de set.
   - Dans le cas où un type de set est déclaré avec OWNER IS SYSTEM, MEMBER MANDATORY AUTOMATIC, lespropriétés SEARCH KEY ou DUPLICATES NOT ALLOWED FOR sont considérées comme des identifiants ou clé d'accès du type member de ce type de set (voir 3.2.4.a. et 3.2.4.g.)

4) Transformation de la clause ORDER IS (ALWAYS) ...

   - FIRST/LAST : c'est l'ordre antichronologique/chronologique par rapport à un owner. Ces ordres sont utiles aux utilisateurs car ils représentent la propriété temporelle de l'ordre d'insertion.
   - NEXT/PRIOR : ce sont des ordres d'insertion par rapport au "courant" du set, par conséquence, on peut camoufler ces propriétés.
   - (ALWAYS) : Dans DML-SPHINX, il n'y a pas de primitive qui permet de changer l'ordre d'un set, ce qui entraîne le camouflage de cette propriété.

5) Simplification de la clause ORDER IS SORTED WITHIN RECORD-NAME

Cette propriété permet de séparer les records member des différents types member lors d'un tri mais quand il y a plusieurs types member nous les transformons en plusieurs types de set différents. Ce qui permet de négliger cette caractéristique s'il n'y a pas d'implications sémantique.

6) Transformation des ordres de tri.

Tous les ordres de tri CODASYL (par des valeurs croissantes ou décroissantes des items associés au type member) sont nécessaires aux utilisateurs DMLP pour manipuler des données d'une BD. Il en est de même du tri par DATABASE KEY car les primitives DML-SPHINX (et ceux de CODASYL) admettent certaines manipulations des données de type DATABASE-KEY.

Les déclarations des clauses de tri ont certaines implications sémantiques que nous établissons comme suit :

CODASYL	MODELE SIMPLIFIE
ORDER IS SORTED	tri par DATABASE-KEY
ORDER IS SORTED ASC/DESC.	tri par ordre croissant /décroissant
ORDER IS SORTED WITHIN RECORD-NAME	tri par DATABASE-KEY
ORDER IS SORTED BY DATABASE-KEY	tri par DATABASE-KEY
ORDER IS SORTED WITHIN RECORD-NAME ... ASC/DESC KEY ...	tri par ordre croissant/décroissant

7) Transformation de l'option DUPLICATES ARE [FIRST / LAST / NOT] ALLOWED dans la clause ASCENDING/DESCENDING KEY.

- DUPLICATES ARE NOT ALLOWED est transformé en un identifiant
- DUPLICATES ARE FIRST/LAST ALLOWED : cette information est utile pour l'utilisateur, nous la conservons comme telle.
- DUPLICATES ARE ALLOWED : C'est inverse de DUPLICATES NOT ALLOWED; ce qui nous permet de simplifier cette option.

8) L'effet et la simplification de la clause MEMBER IS ... MANDATORY/OPTIONAL AUTOMATIC/MANUAL (LINKED TO OWNER)

Les effets des spécifications MANDATORY/OPTIONAL et AUTOMATIC/MANUAL sont comme suit :

a) Pour les primitives CODASYL

	AUTOMATIC	MANUAL
MANDATORY	INSERT : non REMOVE : non	INSERT : oui REMOVE : non
OPTIONAL	INSERT : oui REMOVE : oui	INSERT : oui REMOVE : oui

Quand un type member est déclaré MANDATORY, pour changer un record member d'un set à l'autre de même type, on doit utiliser l'ordre MODIFY

b) Pour les primitives DML de type SPHINX

	AUTOMATIC	MANUAL
MANDATORY	ATTACH : non DETACH : non	ATTACH : oui DETACH : non
OPTIONAL	ATTACH : oui DETACH : oui	ATTACH : oui DETACH : oui

Quand un type member est déclaré MANDATORY, pour transférer un record member d'un set à l'autre du même type, on doit utiliser l'ordre TRANSFER.

LINKED TO OWNER : propriété de performance, l'utilisateur n'a pas besoin de le connaître.

9) Simplification de la clause SET OCCURRENCE SELECTION (SOS) : la SOS de CODASYL a pour but de sélectionner un ou plusieurs set en vue :
   - d'insertion d'un ou plusieurs member automatique au cours d'une exécution d'ordre STORE.
   - de transfert un record d'un set dont il est member obligatoire (MANDATORY) dans un autre du même type.
   - d'accès à un record member désiré lors d'une exécution d'ordre FIND (format-6).

Dans le premier cas, la sémantique de l'ordre CREATE du DML-SPHINX exige que toutes les relations d'accès de type member automatique figurent dans cet ordre avec les étiquettes désignant les courants des types de

record associés, ce qui signifie que tous les records nécessaires pour établir des liens d'accès sont déjà disponibles.

Dans le deuxième cas, la sémantique de l'ordre REACH via un type de set ou TRANSFERT du DML-SPHINX demande qu'une origine (cible) existe déjà.

Pour s'assurer que dans toutes les circonstances, les arguments (items) de sélection ayant une valeur avant l'exécution de SOS, l'ordre REACH de DML-MCS impose une lecture de tous les items associés à un type de record (il n'y a que le GET ALL).

Pour assurer que dans toutes les circonstances les arguments de sélection (items déclarés dans la clause USING et/ou ALIAS de la description de SOS) se trouvent dans des types de record d'un subschéma, nous devons les vérifier; si un des items déclarés dans la clause USING et/ou ALIAS n'appartient pas à un des types de record du subschéma spécifié nous devons le signaler aux utilisateurs ; il en est de même pour les clés de tri.

Les conditions d'exécution d'une SOS sont donc toujours remplies, lors de l'exécution d'un programme DML , en conséquence, on peut la négliger.

a) Notion d'identifiant de set

L'identifiant de set (d'un type donné) est un ensemble d'items associés au type member, tel que chaque valeur de cet ensemble est unique parmi ses valeurs enregistrées dans des records member de ce set. Un type de set peut avoir plusieurs identifiants.

Correspondance

Sauf dans le 2ème cas discuté en 3.2.3.c. (identifiant de type de record), un type de set a un identifiant quand dans la déclaration du type de set on a une des clauses :

- DUPLICATES NOT ALLOWED FOR ...
- ORDER IS SORTED ... { ASCENDING / DESCENDING } KEY IS ... DUPLICATES NOT ALLOWED
- SERACH KEY IS ... DUPLICATES NOT ALLOWED

C'est à dire, on cherche la clause "DUPLICATED NOT ALLOWED". (En général, "DUPLICATES NOT ALLOWED" représente le caractère d'un identifiant, soit global

soit par rapport à un set, soit par rapport à un aréa).

Les utilisateurs doivent connaître cette clause pour faire exécuter leurs programmes correctement.

Un type de set CODASYL peut être déclaré avec plusieurs clauses DUPLICATES NOT ALLOWED, en conséquence, on peut avoir plusieurs identifiants de set.

Remarque :

Dans le cas où un des items constituant l'identifiant (de type de record ou de set) ne se trouve pas dans un subschéma, nous devons le signaler aux utilisateurs.

b) Notion de clé d'accès de set

Une clé d'accès de set est un ensemble d'items associés au type member, tel qu'il existe un moyen de retrouver rapidement à partir d'un set de ce type et d'une valeur de cet ensemble éventuellement un ou plusieurs records members. Un type de set peut avoir plusieurs clés d'accès.

Correspondance et justification

1) Lorsqu'un type de set non singulier ou singulier mais pas fort (MANDATORY AUTOMATIC) est déclaré avec une des clauses :
   - SEARCH KEY ...
   - ORDER IS SORTED INDEXED ... { ASCENDING / DESCENDING } KEY ...

   Nous la transformons en une clé d'accès. En effet, dans CODASYL, on peut déclarer un moyen d'accès aux records member grâce à la clause SEARCH KEY ou à la déclaration d'un set indexé (ORDER IS SORTED INDEXED ...)
2) Dans le rapport CODASYL-71, l'option RANGE de la clause ASCENDING/DESCENDING KEY permet de déclarer un index à plusieurs niveaux et on peut l'utiliser pour décrire un chemin d'accès et spécifier un mécanisme d'accès afin de l'employer éventuellement dans le format-6 de l'ordre FIND. La sémantique du DML-SPHINX n'admet pas une telle utilisation, ce qui permet d'ignorer cette option.

3.3. *Classification des relations d'accès*

1. Relations d'une aréa vers un type de record
2. Relations entre types de record
3. Relations d'un type de record vers des items associés
4. Relations d'un item ou plusieurs items vers un type de record
5. D'autre part, on peut accéder à un record par une variable de travail ou un item de type DATABASE-KEY.

3.4. *Description d'une base de données*

Un schéma d'une base de données est la description complète de cette base de données. Il inclut les descriptions de toutes les aréas, de tous les types de record et des items associés, de tous les types de set existant dans cette BD.

Un subschéma d'un schéma donné est la description partielle d'une base de données. Il se compose de la description des aréas, des types de record, des items associés et des types de set appartenant à ce subschéma.

L'utilisateur ne peut accéder qu'à la partie de BD décrit par un subschéma.

Dans le SGBD-CODASYL, on ne peut travailler que dans la partie de BD décrite par un subschéma (implicitement ou explicitement). En conséquence, nous informons l'utilisateur DML-MCS de la description de subschéma seulement.

Cette description est explicite à partir des modifications des propriétés déclarées dans le subschéma concerné (RENAMING SECTION, modification des chemins d'accès de set, modification des formats des valeurs d'items ...) si elles existent et des propriétés déclarées dans le schéma correspondant, s'il n'y a pas de modification.

Cela nous permet de modifier la définition du subschéma dans le modèle Codasyl simplifié.

3.5. *Contrôle d'autorisation*

Pour accéder ou mettre à jour une BD, il existe éventuellement un contrpole d'autorisation associé à un ordre DML , c'est l'administrateur de BD qui donne des autorisations aux utilisateurs en leur fournissant des informations nécessaires. Ici, nous indiquons les contrôles d'autorisation qui peuvent être associées aux ordres DML .

Type d'information	Opération	Mode d'usage
AREA	OPEN	RETRIEVAL UPDATE EXCLUSIVE RETRIEVAL PROTECTED RETRIEVAL EXCLUSIVE UPDATE PROTECTED UPDATE
Type de record	ATTACH DETACH CREATE SUPPRESS SUPPRESS MODIFY TRANSFER REACH GET	 ONLY SELECTIVE ALL
Type de données élément.	GET MODIFY	
Type de set	ATTACH DETACH REACH	
Sub-schéma	OPEN	

TABLE 3.1.

3.6. *Remarque*

Comme nous l'indiquons au chapitre II, des traitements particuliers (algorithmes de recherche, calcul des DATABASE-KEY...) présentés sous formes de procédures et ajoutés aux procédures de gestion standard d'un SGBD. Ces traitements sont les caractéristiques internes d'un SGBD ou d'un schéma et ont pour but de résoudre les problèmes de performance, d'intégrité, de contrôle d'autorisation ou de consistance ...

Les utilisateurs n'ont pas besoin de connaître.

3.7. *Conclusion*

En général, les simplifications des spécifications codasyl consistent à :

- laisser au compilateur DML-MCS le soin de gérer certains problèmes (SOS avec ALIAS, AREA-ID, LOCATION MODE DIRECT, ...)
- ignorer les propriétés "physique" (mode d'organisation de set, ENCODING/DECODING, ACTUEL/VIRTUAL ...) ou les spécifications s'adressant surtout à l'administrateur (procédures, détermination des mécanismes d'accès ...)
- grouper les différentes spécifications en des notions communes (identifiant, clé d'accès).

D'autre part, il y a des transformations des propriétés pour se conformer avec le DML-SPHINX associé au modèle d'accès : l'item fictif "aréa", l'identifiant, un type de set a un type owner et un type member.

En plus, nous devons faire un compromis entre la simplicité d'un modèle et la puissance du SGBD-CODASYL dans des cas comme :

- le problème de LOCATION MODE DIRECT
- la possibilité de modifier des valeurs d'items ou des records.

° ° °

IIème PARTIE :

DDL DU MODELE SIMPLIFIE

CHAPITRE IV : DDL DU MODELE CODASYL SIMPLIFIE

Dans ce chapitre, nous définissons un langage décrivant le modèle codasyl simplifié. Il s'agit d'un langage utilisé par l'ordinateur pour informer l'utilisateur des descriptions des subschémas du modèle simplifié. Nous rappelons quelques sémantiques des clauses du DDL, les analyses et les significations de clauses ont été présentées au chapitre III. La description des items, en général, suit les règles de Cobol, nous ne les rappelons pas ici.

4.1. Programme DDL

Syntaxe

<déclaration-subschéma>[<déclaration-aréa>]$_1^i$ [<déclaration-type-record>]$_1^j$
[<déclaration-type-set>]$_1^k$ END-SUB-SCHEMA.

Sémantique

Un programme DDL définit un nouveau subschéma (d'un schéma donné) que l'utilisateur peut utiliser.

Les nombres maximums i, j, k sont des limites d'une implémentation.

4.2. Déclaration d'un sub-schéma d'un schéma

Syntaxe

SUB-SCHEMA NAME is <sub-schéma-name> OF SCHEMA <schéma-name>

Sémantique

Déclaration d'un sub-schéma d'un schéma.
Le nom du schéma est unique dans le SGBD, le nom du sub-schéma est unique relativement à ce schéma.

4.3. Déclaration d'un aréa

Syntaxe

AREA NAME IS <area-name>

Sémantique

C'est une aréa d'un subschéma donné, un subschéma a une ou plusieurs aréas. Les aréas d'un subschéma découpent la base de données en des zones

auxquelles les utilisateurs peuvent accéder.

L'utilisation d'une aréa est partageable entre plusieurs programmes. Il y a 6 modes d'usage :

RETRIEVAL : utilisation en mode de lecture, avec partage en mode UPDATE ou PROTECTED UPDATE

UPDATE : utilisation en mode de mise à jour avec partage en mode UPDATE ou RETRIEVAL

PROTECTED RETRIEVAL : avec partage et protection contre une mise de jour des autres programmes

EXCLUSIVE RETRIEVAL : utilisation en mode de lecture, sans partage

PROTECTED UPDATE : avec partage mais protection contre une autre mise à jour

EXCLUSIVE UPDATE : utilisation en mode de mise à jour mais sans partage.

4.4. *Déclaration d'un type de record et des items associés*

Syntaxe

```
RECORD NAME IS <record-name>[ RETRIEVAL ONLY]
    [ IDENTIFIER IS <item-name-1>[ item-name-2>] 1,o ]m,o
    [ ACCESS-KEY IS <item-name-3>[ <item-name-4>] n,o] p,o
    02    AREA PICTURE IS X(n) VALUE IS <area-name-1>[ OR<aréa-name-2>] x,o
    [<level-number><data-name>
       [ PICTURE is <character-string>]
       [USAGE is{COMPUTATIONAL                                  }]
                {COMPUTATIONAL-n                                }
                {DISPLAY                                        }
                {DATABASE-KEY [ OF<record-name-1>]              }
       [SIGN IS{LEADING } [SEPARATE CHARACTER]                  ]
               {TRAILING}
       [{OCCURS <integer-1>TIMES [ INDEXED BY<index-1>[ <index-2>] q,o]                    }]
        {OCCURS <integer-2> TO <integer-3> TIMES DEPENDING ON <item-name-1>}
                                [ INDEXED BY <index-3>[ <index-4>] r,o]
       [{IS RESULT OF PROCEDURE                                                          }]
        {     [ USING <item-name-2>[ <item-name-3>] s,o                                  }
        {     [ ON MEMBER <record-name-2> OF <set-name>] t,o                             }
        { SOURCE IS <item-name-4> OF OWNER OF <set-name-2>                               }
```

```
[ [ CHECK IS [ RANGE OF <literal-1> THRU <literal-2>                    ] ]
             [ PROCEDURE [USING [< item-name-5>]u1]                     ]
[ [ 88< condition-name > [ VALUE IS  [< literal-3> [ THRU <literal-4>]v1] ] ]
                         [ VALUES ARE]
```

Sémantique

La clause < déclaration-type-record > permet de déclarer un type de record ainsi que ses items ou items composés.

Un type de record a au moins un item qui est son identifiant d'aréa (AREA) et a éventuellement un ou plusieurs IDENTIFIER et/ou ACCESS-KEY

4.4.1. La clause RECORD NAME

Syntaxe :

RECORD NAME IS <record-name>[RETRIEVAL ONLY]

Sémantique

1. On désigne un type de record par un nom
2. Certains types de record ne peuvent pas faire l'objet de modifications.

4.4.2. La clause IDENTIFIER

Syntaxe :

[IDENTIFIER IS <item-name-1> [, <item-name-2>]$_{0}^{i}$] $_{0}^{j}$

Sémantique

1. C'est un identifiant global de ce type de record; il est constitué par des items associés
2. Avant la création d'un nouveau record, il faut garnir les valeurs d'items des identifiants.
3. Un type de record peut avoir plusieurs identifiants.

4.4.3. La clause ACCESS-KEY

Syntaxe :

[ACCESS-KEY IS <item-name-1>[, <item-name-2>]$_{0}^{i}$]$_{0}^{j}$

Sémantique

1. C'est une clé d'accès de ce type de record, elle est constituée par des items associés

2.3. voir 4.4.2., sémantique 2.3.

4.4.4. La clause AREA

Syntaxe :

02 AREA PICTURE X(n) VALUE IS <area-name-1> [OR <aréa-name-1>]$_{o}^{*}$

Sémantique

AREA est un item fictif, ayant pour but d'identifier l'aréa d'un record. Les valeurs possibles de AREA d'un type de record sont des noms d'aréas cités dans la clause VALUE.

4.4.5. La clause USAGE

Syntaxe :

$$\text{USAGE IS} \left\{ \begin{array}{l} \text{COMPUTATIONAL} \\ \text{COMPUTATIONAL-n} \\ \text{DISPLAY} \\ \text{DATABASE-KEY [OF <record-name>]} \end{array} \right\}$$

Sémantique

1. En général, c'est la clause USAGE du COBOL
2. DATABASE-KEY OF <record-name> : un item est déclaré comme une référence à un seul type de record désigné par <record-name> .

4.4.6. La clause RESULT

Syntaxe :

RESULT OF PROCEDURE
[USING <item-name-2>[, <item-name-3>]$_{o}^{i}$]
[ON MEMBER <record-name-2> OF <set-name-1>]$_{o}^{j}$

Sémantique

1. Les items indiqués sont des paramètres d'une procédure du SGBD. Ils sont des items du même type record ou ceux des types member indiqués dans ON MEMBER.
2. Utilisateurs ne peuvent pas modifier des valeurs des items défini avec la clause RESULT. Leurs valeurs sont fournies par l'exécution d'une procédure du SGBD.

4.4.7. La clause SOURCE

Syntaxe :

SOURCE IS <item-name-4> OF OWNER OF <set-name-2>

Sémantique

1. Déclarer que les valeurs de cet item (constituant du type member) sont égales à celles d'un item <item-name-4> qui est un constituant du type owner du type de set <set-name-2>
2. Si une réalisation du type member (contenant cet item) n'appartient pas à un set du <set-name-2>, la valeur de cet item dans cette réalisation est nulle.

4.4.8. La clause CHECK

Syntaxe

```
CHECK IS ‖ RANGE OF <literal-1> THRU <literal-2>                    ‖
         ‖ PROCEDURE [USING [ <item-name-5>]ⁱ₁]                     ‖
```

Sémantique

1. Déclarer un contrôle de validité des valeurs d'un item : chaque modification de ses valeurs ou chaque addition du record concerné déclenche un contrôle (soit par RANGE, soit par une procédure du SGBD)
2. RANGE : une valeur d'item déclaré avec clause CHECK doit se trouver entre <litéral-1> et <litéral-2>
3. PROCEDURE : les valeurs de cet item sont vérifiées par une procédure de SGBD
4. Si RANGE ET PROCEDURE sont spécifiés, tous les deux, dans la clause CHECK, RAGNE est exécuté la première.

4.5. *Déclaration d'un type de set*

Syntaxe

```
SET NAME IS <set-name>
OWNER IS { <record-name-1> }
         { SYSTEM          }
[ MEMBER IS <record-name-2> { MANDATORY } { AUTOMATIC }                    ]
                            { OPTIONAL  } { MANUAL    }
[ IDENTIFIER OF SET IS <item-name-1> [ , <item-name-2> ]ⁱ₀ ]ʲ₀
[ ACCESS-KEY OF SET IS <item-bame-3> [ , <item-name-4> ]ᵏ₀ ]ˡ₀
```

```
ORDER IS { FIRST }
         { LAST  }

ORDER IS SORTED | BY DATABASE-KEY
                | { ASCENDING } { KEY IS <item-name-5>[,<item-name-6>] } { k }
                | { DESCENDING }                                       { o }
                |  [DUPLICATES ARE { FIRST } ALLOWED]
                |                  { LAST  }
```

Sémantique

1. La clause <déclaration-type-set> permet de déclarer un type de set. (une relation d'accès entre 2 types de record). Un type de set a un type de record owner et un type record member
2. Un type de set "dynamique" est celui qui n'a pas de type member explicite.

4.5.1. La clause SET NAME

Syntaxe

```
SET NAME IS <set-name>
```

Sémantique

1. Déclaration d'un nom de type de set
2. On peut avoir plusieurs types de set de même nom
3. Un type de set a un type owner et un type member. On distingue un type de set par son nom, celui de son type owner et de son type member
4. Un type de record peut être owner d'un ou plusieurs type de set. De même un type de record peut être member d'un ou plusieurs type de set
5. Un type owner ne peut être un type member dans un même type de set.

4.5.2. La clause OWNER

Syntaxe

```
OWNER IS { <record-name-1> }
         { SYSTEM          }
```

Sémantique

1. <record-name-1> est un type de record déclaré dans ce subschéma
2. SYSTEM : est un type owner fictif généré et géré par SGBD. Un type de set dont l'owner est SYSTEM a un seul set. (c'est un type de set singulier).

4.5.3. La clause MEMBER

Syntaxe

MEMBER IS <record-name-2> { MANDATORY / OPTIONAL } { AUTOMATIC / MANUAL }

Sémantique

1. Déclaration d'un type member
2. <record-name-2> est un type de record déclaré dans ce subschéma
3. AUTOMATIC : déclaration d'une insertion automatique dans un set (de ce type) lors d'un ordre de création d'un record member
4. MANUAL : inversément, on doit utiliser l'ordre ATTACH pour insérer un record du type member dans un set
5. MANDATORY : on ne peut pas utiliser l'ordre de suppression de lien d'accès DETACH. Un record member appartient toujours à un set du type spécifié durant toute sa vie.
6. OPTIONAL : inversément, on peut utiliser l'ordre DETACH

4.5.4. La clause IDENTIFIER OF SET

Syntaxe :

IDENTIFIER OF SET IS <item-name-1> [, <item-name-2>] $^{i}_{o}$

Sémantique

1. Déclaration d'un identifiant d'un type de set
2. <item-name-1>, <item-name-2> : ce sont des items du type member
3. un type de set peut avoir plusieurs identifiants

4.5.5. La clause ACCESS-KEY OF SET

Syntaxe :

ACCESS-KEY OF SET IS <item-name-3> [, <item-name-4>] $^{k}_{o}$

Sémantique

1. Déclaration d'une clé d'accès unique ou multiple d'un type de set
2. <item-name-3>, <item-name-4>: ce sont des items du type member
3. Un type de set peut avoir plusieurs clés d'accès.

4.5.6. La clause ORDER

Syntaxe:

{ ORDER IS { FIRST / LAST }

ORDER IS SORTED { BY DATABASE-KEY

{ ASCENDING / DESCENDING } KEY IS <item-name-5> [, <item-name-6>]

[DUPLICATES ARE { FIRST / LAST } ALLOWED } }

Sémantique

1. Déclaration d'un ordre d'un type de set
2. FIRST : ordre antichronologique
3. LAST : ordre chronologique
4. SORTED BY DATABASE-KEY : ordre de tri par DATABASE-KEY
5. SORTED ASCENDING/DESCENDING KEY : ordre de tri selon des valeurs croissantes/décroissantes des items (< item-name-5 > , <item-name-6 >) du type member

CHAPITRE V : DBMS-20 ET DDL DU MODELE SIMPLIFIE

5.1. DBMS-20

DBMS-20 est basé sur le rapport CODASYL-71 et implémenté sur DEC-20. DBMS-20 a choisi les langages COBOL et FORTRAN comme langages hôtes du DML, dans le cadre de notre travail, ce sont les aspects COBOL qui nous concernent seulement.

Les spécifications de DBMS-20 sont plus simples que celles du CODASYL-71, dans ce chapitre nous citons les descriptions d'un schéma et d'un subschéma DBMS-20 associé et nous indiquons les restrictions afin de définir le DDL du modèle simplifié correspondant. La sémantique du DDL-DBMS-20 est présentée dans B10 (*)

5.1.1. Le DDL du DBMS-20

Remarquons qu'un nom est représenté par une chaîne de caractères alphanumériques et le tiret, le premier caractère doit être alphabétique. La désignation des noms d'items élémentaires ou composés suit les règles de COBOL.

1. Schéma

Déclaration d'un schéma :

Syntaxe

SCHEMA NAME IS schéma-name.

Restrictions

1. La longueur du nom d'un schéma est de 6 caractères au maximum
2. Il n'y a pas de contrôle d'autorisation.

2. Aréa

Déclaration d'une aréa :

(*) voir bibliographie

Syntaxe

```
AREA NAME IS area-name-1 [AREA IS TEMPORARY]

[ PRIVACY LOCK FOR { EXCLUSIVE } { UPDATE    } IS lock-1 ]
                   { PROTECTED } { RETRIEVAL }                .
```

Restrictions

1. La longueur de nom d'une aréa est de 30 caractères au plus
2. Il y a éventuellement un contrôle d'autorisation pour les modes d'usage. Dans DBMS-20, il n'y a que des contrôles d'autorisation au niveau d'aréa et subschéma.

3. Type de record et des items (élémentaires ou composés) associés

Déclaration d'un type de record

Syntaxe

```
RECORD NAME IS record-name-1
  LOCATION MODE IS { DIRECT identifier-1                          }
                   { CALC USING data-name-1 [data-name-2]         }
                   {   [DUPLICATES NOT ALLOWED]                   }
                   { VIA set-name-1                               }
  WITHIN area-name-1 [area-name-2 ... AREA-ID is identifier-2]
02 data-name-3 { PICTURE ... } [OCCURS ...].
               { SIZE ...    }
               { TYPE ...    }
```

Restrictions

1. Longueur maximum de record name, identifier, data-name 30 caractères (identifier-i est une variable de travail)
2. Dans un schéma, il y a 480 types de record au plus
3. Un item associé à un type de record ne peut pas être déclaré dans la clause LOCATION MODE IS DIRECT
4. On ne peut pas déclarer une procédure particulière associée à LOCATION MODE CALC.
   En général, toutes les procédures du DBMS-20 sont standardisées, il n'y a pas de possibilité de déclarer des autres procédures, en conséquence, il n'y a pas de clauses "ON".
   data-name-1, data-name-2 sont des items élémentaires (par abréviation items) associés.

Déclaration d'items (élémentaires ou composés) associés

Syntaxe

Format 1

```
02 data-name-3 [%pseudonym-3] { PICTURE } IS picture-string
                              { PIC     }
  [ USAGE IS  DISPLAY   ]
              DISPLAY-6
              DISPLAY-7   ] [OCCURS integer-1 TIMES].
              DISPLAY-9   ]
```

Format 2

```
02 data-name-3 [%pseudonym-3] SIZE IS integer-2 { WORDS                          }
                                                { USAGE { DISPLAY   }            }
                                                        { DISPLAY-6 }            }
                                                        { DISPLAY-7 }            }
                                                        { DISPLAY-9 }            }

  [OCCURS integer-1 TIMES].
```

Format 3

```
02 data-name-3 [%pseudonym-3]  TYPE IS { [ FLOAT ] [ { DECIMAL } ] [ REAL    ] }
                                       { [ FIXED ] [ { DEC     } ] [ COMPLEX ] }
                                       { DBKEY                                 }

  [integer-3] [, integer-4] } [OCCURS integer-1 TIMES].
```

Pour la clarté de l'exposé, nous rappelons quelques significations des descriptions d'items :

Format 1 : décrit un item (élémentaire) alphanumérique

Format 2 : décrit un item composé, les descriptions détaillées se trouvent dans des subschémas.

Format 3 : décrit un item numérique ou une database-key.

Les types sont convertis automatiquement en ceux de ANSI COBOL.

Il y a 3 codes de représentations des valeurs d'items élémentaires ou composés : 6-BIT, 8-BIT (ASCII), 9-BIT (EBCDIC)

Restrictions

1. Format 1 : on ne peut utiliser que des symboles S9AXPV et () dans la clause PICTURE
2. Les vecteurs ou groupes répétitifs doivent être de longueur fixe (ce qui entraîne la longueur fixe des records).
3. Il n'y a pas de clauses : RESULT, SOURCE, CHECK, ENCODING/DECODING

4. Les déclarations des descendants d'un item composé d'un subschéma COBOL suivent les règles du langage COBOL implémenté sur DEC-20. Ces descriptions sont considérées comme un texte.

4. Les types de set

Déclaration d'un type de set

Syntaxe

```
SET NAME IS set-name-1

    MODE IS CHAIN [LINKED TO PRIOR]

    ORDER IS { ALWAYS { FIRST | LAST | NEXT | PRIOR }
             { SORTED [ WITHIN RECORD-NAME
                      | BY DATABASE-KEY
                      | DUPLICATES ARE [ FIRST | LAST | NOT ] ALLOWED ] }

    OWNER IS { record-name-1 | SYSTEM } [.]

    MEMBER IS record-name-1 { MANDATORY | OPTIONAL } { AUTOMATIC | MANUAL }
        [LINKED TO OWNER]
    [ { ASCENDING | DESCENDING | ASCENDING RANGE | DESCENDING RANGE }
          KEY IS data-name-1 [data-name-2] ...
        [DUPLICATES ARE [ FIRST | LAST | NOT ] ALLOWED] ]
    [ SET OCCURRENCE SELECTION IS THRU { CURRENT OF SET
                                       | LOCATION MODE OF OWNER } ]
    [ { USING data-name-2 [data-name-3]. . .
      | ALIAS FOR data-name-4 IS identifier-1 [%pseudonym-1]. . . } ] [.]
```

Restrictions

1. Set-name : chaîne de caractères, de longueur 30 caractères au plus.
2. Mode d'organisation : il n'y a pas de POINTER-ARRAY DYNAMIC (ni de POINTER ARRAY) en conséquence :
   - il n'y a pas de type de set dynamique
   - il y a toujours une relation d'accès inverse entre type owner et type member
3. DML-DBMS-20 n'a pas de primitive ORDER pour changer l'ordre d'un set (en effet, c'est toujours ALWAYS)
4. Il n'y a pas de clause DUPLICATES NOT ALLOWED FOR ... séparément
5. Il n'y a pas de SEARCH KEY, ORDER IS SORTED INDEXED
6. Dans SOS, il n'y a que le format 1(de Codasyl) et la SOS ne sert qu'à sélectionner un set pour insérer un record member automatique
7. Il n'y a pas de format 6 de l'ordre FIND
8. Un type de record peut participer, comme type owner ou type member, à 512 types de set au plus. En effet, on ne peut avoir que 512 pointeurs de chainage des sets au maximum dans un record.

5. Sub-schéma

Déclaration d'un sub-schéma

Syntaxe

```
SUB-SCHEMA NAME IS sub-schema-name
    [ PRIVACY LOCK IS lock-1] .
Format 1
AREA SECTION
    COPY area-name-1 [ area-name-2] ...
Format 2
AREA SECTION.
    COPY ALL AREAS.
Format 3
AREA SECTION.
    COPY TEMPORARY area-name-3 [ area-name-4] ...
```

Format 1

```
RECORD SECTION.

01  record-name-1.
01  record-name-2.
[ 02  data-name-1.                          ]
[   [COPY sub-schema-name TEXT.]            ]
[COPY OTHERS .]
```

Format 2

```
RECORD SECTION.
COPY ALL RECORDS.
```

Format 1

```
SET SECTION
   COPY set-name-1 [ set-name-2] ...
```

Format 2

```
SET SECTION
   COPY ALL SETS.
```

Restrictions

1. LOCK-1 : chaine de caractères alphanumériques de longueur maximum 5 caractères
2. Il y a au maximum 36 sub-schémas donné
3. Dans DBMS-20, on définit un sub-schéma comme un sous ensemble d'un schéma. (En effet, les clauses de descriptions d'un sub-schéma ne sont que des COPY ou des descriptions d'items descendants d'un item composé -DATA AGGREGATE-ou une déclaration d'une aréa "provisoire")
4. Il n'y a éventuellement qu'un contrôle d'autorisation pour compiler un programme utilisant une (partie d'une) BD.
5. Les noms des types d'informations d'un sub-schéma sont les mêmes que ceux du schéma concerné (pas de RENAMING SECTION)
6. Les descriptions des items descendants suivent les règles de COBOL-DBMS-20.

5.1.2. La classification des relations d'accès

1. Relations d'une aréa vers un type de record
2. Relations entre types de record
3. Relation d'un type de record vers des items, élémentaires ou composés, associés.
4. Relations d'un item ou plusieurs items vers un type de record :

Ici, on n'a que des clés d'accès de type de record (CALC)

5. D'autre part, on peut accéder à un record par référence (DATABASE-KEY)

5.2. *DDL du modèle DBMS-20 simplifié*

Les spécifications du DBMS-20 sont moins nombreuses que celles de Codasyl, ce qui entraine la suppression d'un nombre de propriétés du modèle CODASYL simplifié (ces propriétés, présentées dans les clauses du DDL-CODASYL ne sont pas implémentées sur DBMS-20). Nous présentons ci-dessous le DDL du modèle DBMS-20 simplifié ainsi que les correspondances avec DBMS-20, s'il existe des restrictions ou des particularités du système.

Règles relatives aux noms :
Un nom est représenté par une chaine de caractères alphanumériques et le tiret, la première doit être un caractère alphabétique.

La désignation des noms d'items élémentaires ou composés associés à un type de record suit les règles de COBOL.

5.2.1. Programme DDL

Syntaxe

<déclaration-sub-schéma>[<déclaration-aréa>]$_{1}^{i}$ [déclaration-type-record] $_{1}^{480}$
[déclaration-type-set]$_{1}^{j}$ END-SUB-SCHEMA

5.2.2. Déclaration d'un sub-schéma d'un schéma

Syntaxe

SUB-SCHEMA NAME IS <sub-schéma-name> OF SCHEMA <Schéma-name>

5.2.3. Déclaration d'une aréa

Syntaxe

AREA NAME IS <area-name>

5.2.4. Déclaration d'un type de record et des items associés

Syntaxe

RECORD NAME IS <record-name>[RETRIEVAL ONLY]
[IDENTIFIER IS <item-name-1>[<item-name-2>]$_{0}^{i}$]$_{0}^{1}$
[ACCESS-KEY IS <item-name-3>[<item-name-4>]$_{0}^{j}$]

```
02 AREA PICTURE X(30) VALUE IS <area-name-1>[ OR <area-name-2>] k
                                                               o
[{{ 02 <data-name>                                            }}
   {{   [PICTURE  <character-String>]                         }}
   {{   [USAGE { COMP                                         }}
   {{          { COMP-1                                       }}
   {{          { COMP-3                                       }}
   {{          { DISPLAY-6                                    }}
   {{          { DISPLAY-8                                    }}
   {{          { DISPLAY-9                                    }}
   {{          { DBKEY                                        }}
   {{   [OCCURS <integer> TIMES]                              }}
   {{ <text>                                                  }}
```

5.2.4.1. La clause RECORD NAME

Syntaxe

RECORD NAME IS <record-name>[RETRIEVAL ONLY]

Correspondance

1. Il y a 2 cas où on peut accéder aux réalisations d'un type de record pour la lecture seulement :
   - un type de record est déclaré avec LOCATION MODE CALC et l'un des items élémentaires spécifiés ne se trouve pas dans le SUB-SCHEMA
   - un type member déclaré avec SORT-KEY et un des items élémentaires utilisés dans la clé de tri ne se trouve dans le SUB-SCHEMA spécifié

   En conséquence, il faut vérifier ces deux cas.
2. Lorsqu'un chemin d'accès (SOS) utilise des valeurs d'items, ces items doivent être déclarés dans une des 2 clauses :

   LOCATION MODE CALC ... DUPLICATES NOT ALLOWED

   {ASCENDING } KEY IS ... DUPLICATES NOT ALLOWED
   {DESCENDING}

   Ce qui justifie que l'examen de ces 2 cas est suffisant.

5.2.4.2. La clause IDENTIFIER

Syntaxe

IDENTIFIER IS <item-name-1> [<item-name-2>] i
                                            o

Correspondance

Ici, il n'y a que deux cas :

- un type de record est déclaré avec LOCATION MODE IS CALC ...DUPLICATES NOT ALLOWED
- Un type de set dont type owner est SYSTEM et type member est MANDATORY AUTOMATIC avec un tri :

  {ASCENDING } [RANGE] KEY ... DUPLICATES NOT ALLOWED
  {DESCENDING}

5.2.4.3. La clause ACCESS-KEY

Syntaxe

[ACCESS-KEY IS <item-name-3> [<item-name-4>] $\begin{matrix} j \\ o \end{matrix}$]

Correspondance

Ici, il n'y a qu'un seul cas possible : lorsque un type de record est déclaré LOCATION MODE IS CALC . En conséquence, il n'y a qu'une seule clé d'accès. car :

- Dans DBMS-20, il n'y a pas de SEARCH KEY ou de la clause ORDER IS SORTED INDEXED
- Il n'y a pas de Format 6 de l'ordre FIND (de même, SOS ne sert qu'à sélectionner un set en vue d'insertion d'un record member automatique).

Ce qui justifie la restriction précédente.

5.2.4.4. La clause USAGE

Syntaxe

USAGE { COMP | COMP-1 | COMP-3 | DISPLAY-6 | DISPLAY-7 | DISPLAY-9 | OBKEY }

Correspondance

1. C'est la clause USAGE déclarée dans le sub-schéma spécifié ou dans le schéma correspondant.
2. DISPLAY est équivalent à DISPLAY-6.
3. Quand un item numérique est déclaré sous le format 3, nous faisons une transformation en clause USAGE-COBOL
4. Le format-2 permet de déclarer un item composé dans un schéma, sa description doit être décrite dans un sub-schéma de ce schéma. Ici, nous faisons les références au niveau du sub-schéma.
5. La conversion entre items numériques d'un schéma et ceux d'un sub-schéma correspondant.

Schema Declaration	Precision Range (Bits)	Default Precision (Bits)	COBOL type
FIXED BIN REAL	<36	35	COMP PIC S9 (1-10)
FIXED BIN REAL	36-70	---	COMP PIC S9 (11-18)
FLOAT BIN REAL	<28	27	COMP-1
FLOAT BIN REAL	28-62	---	COMP PIC S9 (18)
FLOAT BIN COMPLEX	<28	27	COMP PIC S9 (18)
FIXED DEC REAL	<19	10	COMP-3 PIC S9 (prec)

Relation between Binary and Decimal Precision

Binary Precision declared in Schema	Decimal Precision in COBOL
1-4	PIC S9 (1)
5-7	PIC S9 (2)
8-10	PIC S9 (3)
11-14	PIC S9 (4)
15-17	PIC S9 (5)
18-20	PIC S9 (6)
21-24	PIC S9 (7)
25-27	PIC S9 (8)
28-30	PIC S9 (9)
31-35 (default)	PIC S9 (10)
36-38	PIC S9 (11)
39-41	PIC S9 (12)
42-44	PIC S9 (13)
45-48	PIC S9 (14)
49-51	PIC S9 (15)
52-54	PIC S9 (16)
55-58	PIC S9 (17)
59-70	PIC S9 (18)

5.2.4.5. La clause OCCURS

Syntaxe

OCCURS <integer> TIMES

Correspondance

1. C'est la clause OCCURS déclarée dans un schéma
2. DBMS-20 n'admet que des groupes répétitifs ou des vecteurs de longueur fixe. Ce qui entraîne que la longueur des records d'un type donné est fixe.

5.2.4.6. La clause <text>

Syntaxe

<text>

Correspondance

Les descriptions des attributs des items composés sont considérées comme un texte. Le compilateur DDL-DBMS-20 ne vérifie pas les règles syntaxiques (COBOL). Les vérifications de ces règles sont effectuées par le compilateur COBOL. Nous laissons comme telles les descriptions des attributs des items composés.

5.2.5. Déclaration d'un type de set

Syntaxe

```
SET NAME IS <set-name>
   OWNER IS { <record-name-1> }
            { SYSTEM          }
   MEMBER IS <record-name-2> { MANDATORY } { AUTOMATIC }
                             { OPTIONAL  } { MANUAL    }
  [ IDENTIFIER OF SET IS <item-name-1> [<item-name-2>] i ]
                                                      o
[[ ORDER IS { FIRST } ]
            { LAST  }
[ ORDER IS SORTED { BY DATABASE-KEY                                                                    } j ]
                  { ASCENDING  } KEY IS <item-name-3> [<item-name-4>]                                  } o ]
                  { DESCENDING } [DUPLICATES ARE { FIRST } ALLOWED]                                    }
                                                 { LAST  }                                             ]]
```

Correspondance

1. IDENTIFIER : dans DBMS-20, il n'y a que la clause DUPLICATES NOT ALLOWED associée à un tri selon des valeurs des items du type member, ce qui entraîne la restriction du nombre d'identifiant de set
2. Il n'y a pas ACCESS-KEY OF SET car dans DBMS-20 il n'y a pas de SEARCH KEY (et de format 6 de l''ordre FIND), et de la clause ORDER IS SORTED INDEXED

3ème PARTIE :

SCHEMAS CONCEPTUELS

CHAPITRE VI : SCHEMA CONCEPTUEL DES TYPES DE DONNEES CODASYL-71

6.1. Introduction

Notre but, dans ce travail, est de générer des listings DDL du modèle DOCASYL simplifié et des tables du compilateur à partir des SCHEMAS d'un SGBD-CODASYL. Nous allons construire une base de données pour ces deux générations. Cette base de données possède son propre schéma; avant de le définir, nous tentons de représenter les types de données (les concepts ou les définitions) CODASYL-71 par un schéma conceptuel.

Ce procédé qui est une démarche classique d'analyse des systèmes d'information se justifie par le nombre de SGBD qui s'inspirent des concepts fondamentaux présentés dans des rapports du DBTG, (DBMS-20, PHOLAS, UDS, IDS, ...)

6.2. Modèle conceptuel des définitions Codasyl

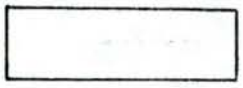
Les spécifications CODASYL sont très nombreuses, notamment les relations entre les différents concepts. Ce qui rend difficile une classification de ces concepts. D'une manière générale, on peut les classer en 7 problèmes :

- la description d'une BD et les descriptions partielles de cette base
- la subdivision d'une BD (aréas)
- les types de record et les items associés
- les relations entre les types de record (types de set)
- la performance
- les contrôles d'intégrité ou d'autorisation
- les valeurs des items et leurs modes de représentation

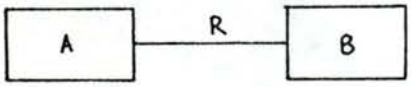
Cependant, une telle classification ne permet d'avoir qu'une première vue des spécifications CODASYL, les concepts CODASYL et les relations entre eux sont plus nombreux et plus compliqués.

En vue de la construction du modèle conceptuel, nous présentons les concepts CODASYL sous forme du modèle relationnel binaire. (voir bibliographie B1). Les définitions CODASYL sont rappelées brièvement à titre indicatif.

Les représentations graphiques : nous utilisons le formalisme suivant :



: un objet complexe désigne un concept, une définition CODASYL.



: représente l'association (relation) R entre les 2 concepts A et B.

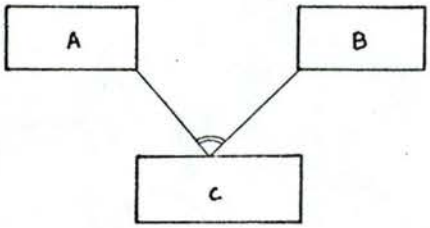
Propriété de cardinalité :

——— : relation "one to one" faible

—◁— : relation "one to many", faible

—▷◁— : relation "many to many", faible.

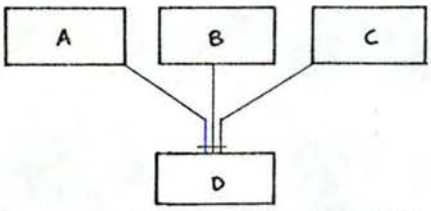
| : (associé à un arc) relation forte du côté marqué par ce signe.



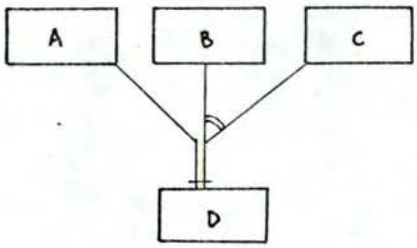
: représente une association :

- soit entre le concept C et A
- soit entre le concept C et B

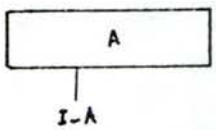
(c'est un "ou exclusif")



: un élément de D est constitué par un élément de A, un élément de B et un élément de C (en terme du modèle relationnel n-aires c'est une relation D(A,B,C))



: un élément de D est constitué :
- soit par un élément de A et un élément de B
- soit par un élément de A et un élément de C



: un objet élémentaire I-A associé à un concept A représente une propriété de ce concept.

D'autre part, une relation entre deux objets A et B est représentée par un arc non orienté (ce qui signifie l'existence d'une association entre A et B), à l'exception d'une relation d'un objet complexe vers lui-même, nous devons le représenter par un arc orienté pour enlever l'ambiguïté.

Les symboles représentant la propriété de cardinalité ne permettent pas de l'exprimer dans certains cas; nous utilisons donc le quadruplet I-J, K-L pour exprimer cette propriété dans ces cas.

Soit R(A, B), la relation R entre deux objets (ou concepts) A et B, sa cardinalité est présentée comme suit :
R(A,B) ≡ I-J, K-L
ce qui signifie qu'un élément de A(B) a au moins I(K) éléments de B(A) et au plus J(L) éléments de B(A).

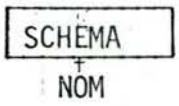
Pour simplifier les graphes nous admettons des relations sans nom. Nous utilisons un même nom désignant plusieurs relations qui représentent une même propriété, un même concept.

6.2.1. Schéma

Schéma est la description complète d'une BD-CODASYL. On le désigne par un nom. Un nom est une chaîne formée de caractères.

DDL-CODASYL
SCHEMA IS schema-name

Graphe correspondant



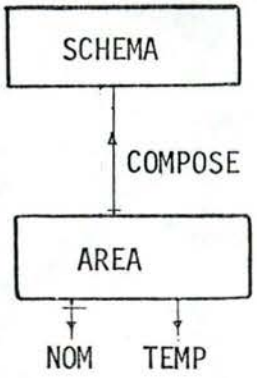
6.2.2. Aréas

Une aréa est une subdivision d'une BD
On la désigne par un nom
Une aréa peut être provisoire ou non (TEMPORARY)

DDL-CODASYL
AREA NAME IS area-name-1
[;AREA IS TEMPORARY]

Graphe correspondant

COMPOSE(SCHEMA, AREA) : un schéma se compose de la description des aréas (et des autres "objets")
(cette relation exprime aussi l'appartenant de l'AREA au SCHEMA)
TEMP : représente la clause AREA IS TEMPORARY



Contrainte d'intégrité

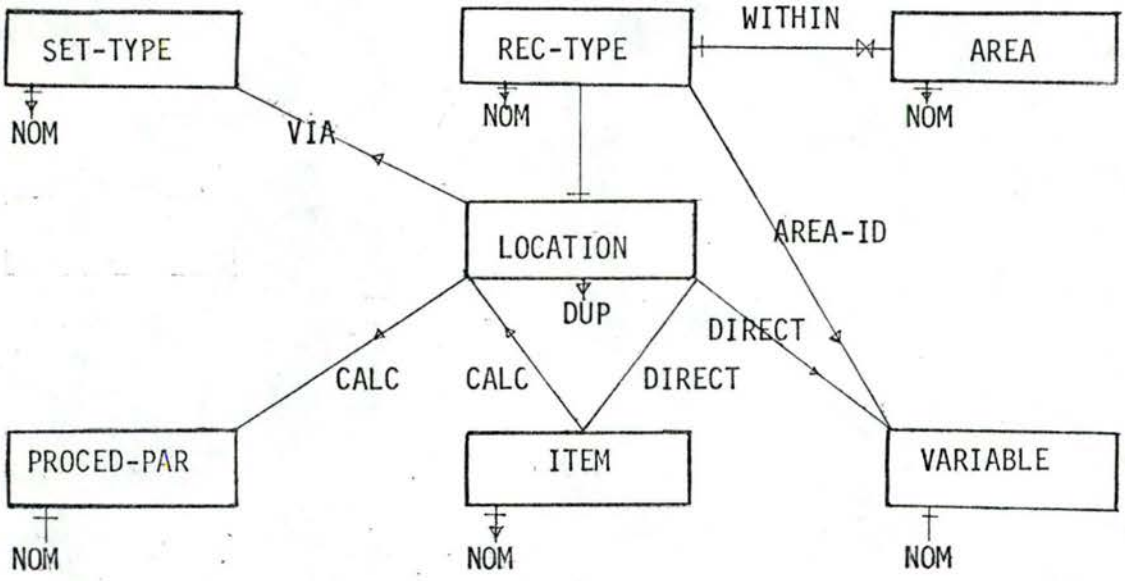
Dans la suite, nous présentons les contraintes d'intégrité qui ne sont pas exprimées dans les graphes.
Le nom d'une aréa doit être unique dans un schéma

6.2.3. Les types de record

Un type de record est un ensemble de type de données qui définissent un même type d'entité , un même type d'individu.
On le désigne par un nom.
DDL-CODASYL

```
RECORD NAME IS record-name-1

[;LOCATION MODE IS { DIRECT { data-base-data-name-1 }
                   {        { data-base-identifier-1 }

                   { CALC [data-base-procedure-1] USING data-base-identifier-2
                   {      [,data-base-identifier-3]... DUPLICATES ARE [NOT] ALLOWED

                   { VIA set-name-1 SET                                          ]

;WITHIN area-name-1 [ {,area-name-2}... AREA-ID IS data-base-data-name-2 ]
```

Graphe correspondant



REC-TYPE représente le concept de type de record

LOCATION représente les modes d'emplacement (la clause LOCATION MODE)

(REC-TYPE, LOCATION) représente la spécification de mode d'emplacement d'un type de record.

DIRECT(LOCATION, VARIABLE) : Le LOCATION MODE DIRECT utilise une variable pour le rangement d'un nouveau record.

DIRECT(LOCATION, ITEM) : le LOCATION MODE DIRECT utilise un item (ITEM) pour le rangement d'un nouveau record

CALC(LOCATION, ITEM) : le LOCATION MODE CALC utilise des valeurs d'items pour effectuer un calcul d'adresse.

Ces valeurs peuvent être unique ou non (LOCATION, DUP)

CALC(LOCATION, PROCED-PAR) :le LOCATION MODE CALC peut utiliser une procédure non standard (PROCED-PAR) pour calculer une adresse à partir des valeurs d'items

VIA(LOCATION, SET-TYPE): le LOCATION MODE VIA utilise un type de set (SET-TYPE)

WITHIN(REC-TYPE, AREA) : la localisation des zônes de la BD auxquelles se trouvent des records d'un type spécifié. (la clause WITHIN)

AREA-ID(REC-TYPE, VARIABLE) : s'il existe plusieurs aréas dans la clause WITHIN, une variable est utilisée pour identifier l'aréa voulue.

Contrainte d'intégrité

1. Le nom d'un type de record doit être unique dans un schéma
2. Lorsqu'un type de record a LOCATION MODE CALC, les items cités dans cette clause doivent être ceux associés à ce même type de record
3. Lorsqu'un type de record a LOCATION MODE VIA, le type de set cité dans cette clause doit exister dans le schéma
4. On a soit la relation VIA soit une des 2 relations DIRECT soit la relation CALC (LOCATION, ITEM) et éventuellement CALC(LOCATION, PROCED-PAR)

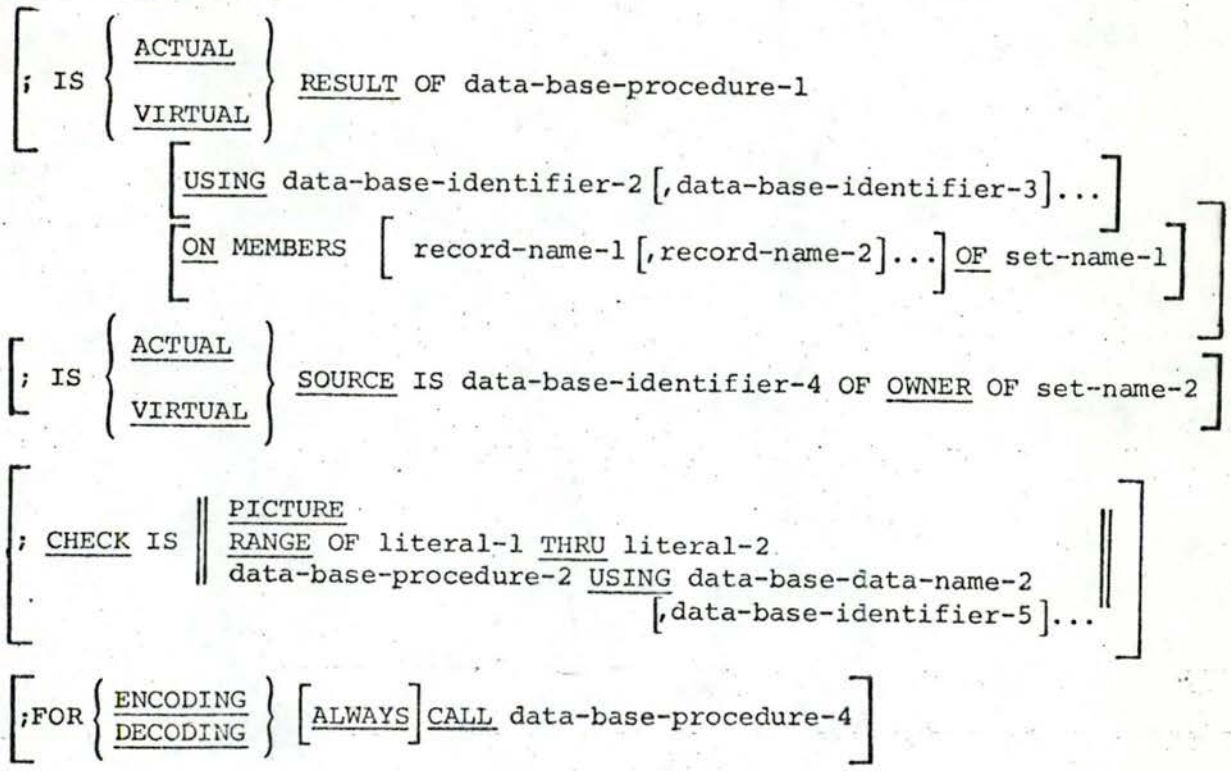
6.2.4. Les items associés

Un item élémentaire (par abréviation item) est le plus petit élément de type de données. Un item composé est un ensemble de valeurs d'items, il y a deux classes : vecteur et groupe répétitif.

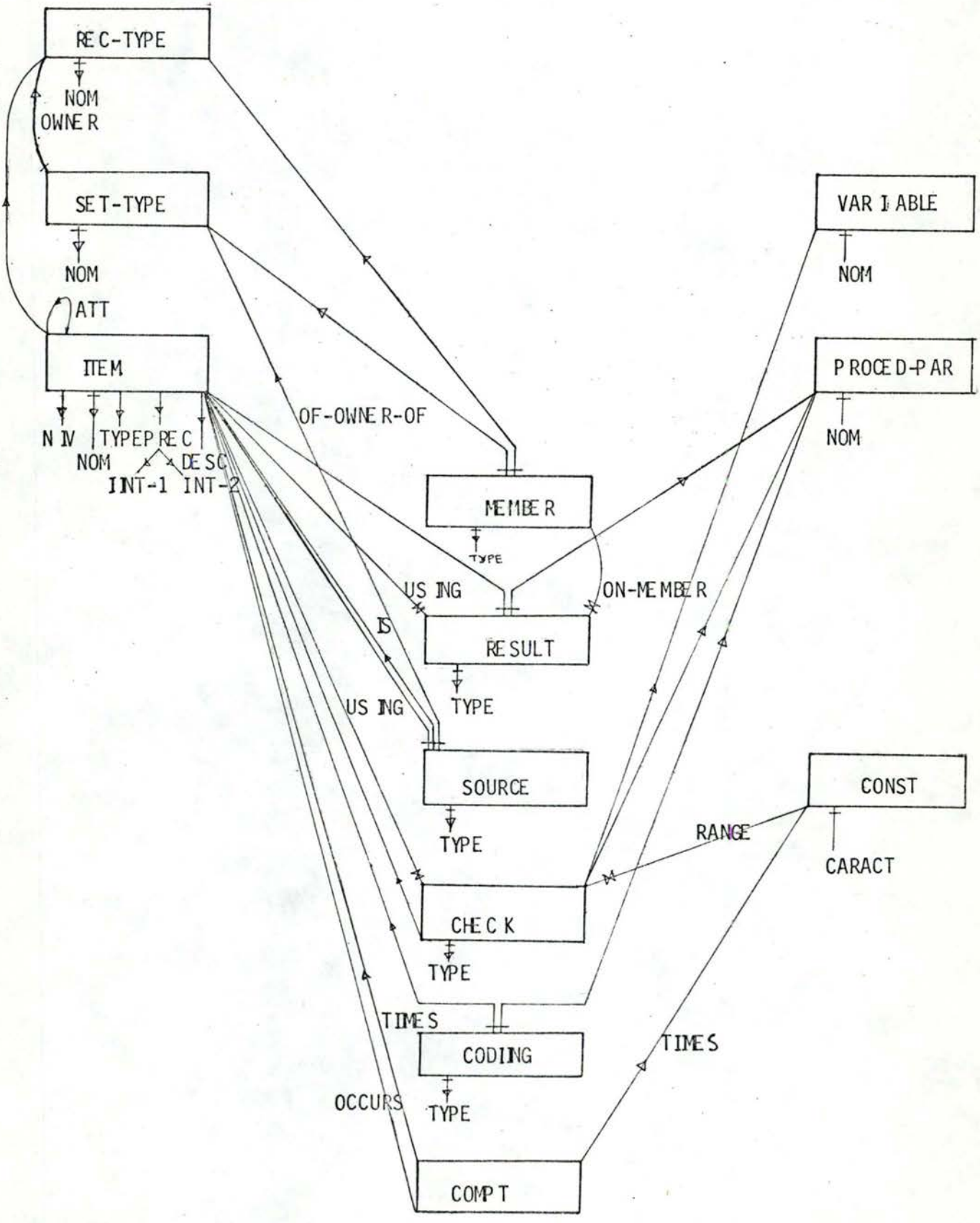
La désignation d'un item se compose d'un nombre de niveau hiérarchique et un nom.

DDL-CODASYL

```
[level-number] data-base-data-name-1

[;PICTURE IS " { character-string-picture-specification } "
              { numeric-picture-specification         }  ]

                                                                  [integer-1 [,integer-2]]
                { BINARY  }
                { DECIMAL }
                { FIXED   }
                { FLOAT   }
                { REAL    }
                { COMPLEX }
[;TYPE IS       { BIT     }                                       [integer-3]              ]
                { CHARACTER }
                DATABASE-KEY
                implementor-type

[;OCCURS { integer-4           } TIMES ]
         { data-base-identifier-1 }
```


; IS ACTUAL VIRTUAL RESULT OF data-base-procedure-1
USING data-base-identifier-2 [,data-base-identifier-3]...
ON MEMBERS [record-name-1 [,record-name-2]...] OF set-name-1
; IS ACTUAL VIRTUAL SOURCE IS data-base-identifier-4 OF OWNER OF set-name-2
; CHECK IS PICTURE
RANGE OF literal-1 THRU literal-2
data-base-procedure-2 USING data-base-data-name-2
[,data-base-identifier-5]...
;FOR ENCODING DECODING [ALWAYS] CALL data-base-procedure-4

Graphe correspondant



ITEM représente les items associés à un type de record (REC-TYPE) par les 2 relations (REC-TYPE, ITEM) et ATT(ITEM, ITEM)

(REC-TYPE, ITEM) : la relation entre un type de record et ses items de niveau 01.

ATT(ITEM, ITEM) : la relation entre un item composé et ses descendants

DESC : désigne la clause PICTURE

TYPE : désigne la clause TYPE

PREC : désigne la précision de valeurs d'un item numérique (INT-1, INT-2) ou le nombre de bits (caractères) d'un item élémentaire (Int-1)
INT-1 représente l'option "integer-1" ou "integer-3"
INT-2 représente l'option "integer-2"

COMPT désigne un compteur pour un vecteur ou un groupe répétitif (clause OCCURS)

RESULT désigne la clause RESULT qui peut utiliser des items (la relation USING(RESULT, ITEM)) et/ou des records member d'un type de set (les deux relations ON-MEMBERS(RESULT, MEMBER) et (MEMBER,SET-TYPE))(x)

MEMBER représente le concept de type member

(RESULT, PROCED-PAR) désigne la procédure qui calcule les valeurs d'un item déclaré avec la clause RESULT

SOURCE : désigne la clause SOURCE

IS(ITEM,SOURCE) : un item peut être "source" d'un ou plusieurs autres items

OF-OWNER-OF(SOURCE,SET-TYPE) désigne la clause "OF OWNER OF set-name-2" de la clause SOURCE. Cette relation permet de retrouver item "source" du type owner (via la relation OWNER)

CHECK : désigne la clause CHECK

RANGE(CHECK,CONST) désigne l'option RANGE OF ... de la clause CHECK (CONST désigne des constantes qui sont des chaînes de caractères).

(CHECK, PROCED-PAR) et (CHECK, VARIABLE) désignent une procédure de contrôle de validité des valeurs d'items et une variable contenant le code

---

(x) En effet, la relation (MEMBER, SET-TYPE) remplace la spécification "OF Set-name" de la clause RESULT.

d'erreur de retour.

USING(CHECK, ITEM) : la procédure de contrôle de validité peut utiliser les valeurs des autres items

CODING désigne la clause ENCODING/DECODING

Contraintes d'intégrité

1. La qualification des noms d'items doit être unique dans un schéma
2. La relation ATT(ITEM, ITEM) est dans un ordre bien déterminé
3. (ITEM,CODING) ≡ 0-2, 1-1
4. (ITEM,RESULT) et (ITEM,SOURCE) sont mutuellement exclusives
5. (ITEM, CHECK) ≡ 0-3, 1-1
6. Les nombres de niveau d'items compris entre 01 et 99
7. Un item élémentaire ou composé doit appartenir à un et un seul type de record.

6.2.5. Les types de set

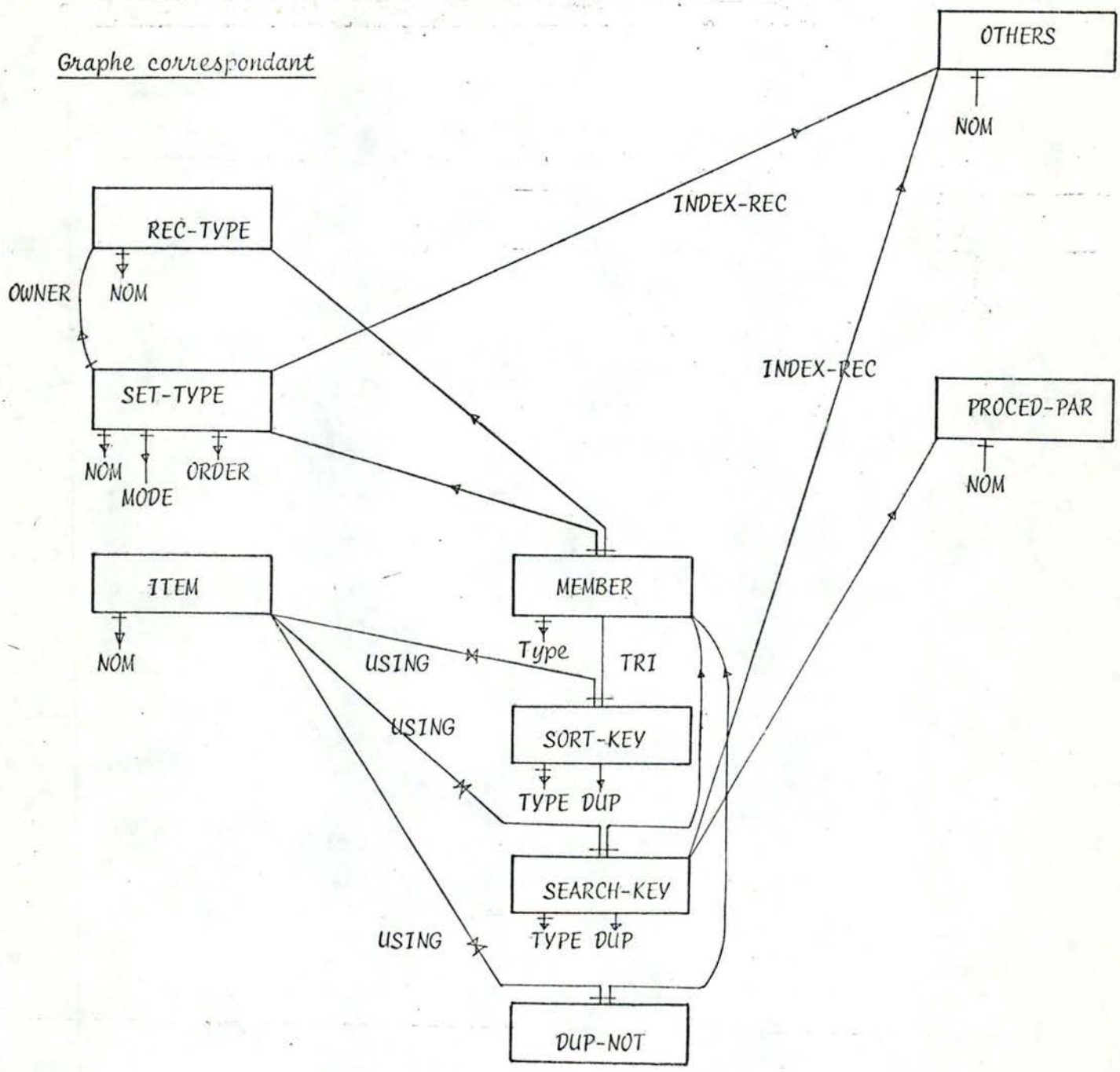
Un type de set est une relation entre un type owner et un ou plusieurs types member.

On le désigne par un nom.

DDL-CODASYL

```
SET NAME IS set-name-1

;MODE IS { CHAIN [LINKED TO PRIOR]                    }
         { POINTER-ARRAY [DYNAMIC]                   }
         { implementor-name                          }

Format 1

;ORDER IS [ALWAYS] { FIRST }
                   { LAST  }
                   { NEXT  }
                   { PRIOR }

Format 2

;ORDER IS SORTED [INDEXED [NAME IS index-name-1]] [ WITHIN RECORD-NAME
                                                   BY DATABASE-KEY
                                                 DUPLICATES ARE [FIRST] ALLOWED
                                                                [LAST ]
                                                                [NOT  ] ]

;OWNER IS { record-name-1 }
         { SYSTEM        } .

MEMBER IS record-name-1 { MANDATORY } { AUTOMATIC } [LINKED TO OWNER]
                        { OPTIONAL  } { MANUAL    }
    [DUPLICATES ARE NOT ALLOWED FOR data-base-identifier-1
                [,data-base-identifier-2]...]...
[;{ASCENDING } [RANGE] KEY IS data-base-identifier-3[,data-base-identifier-4]...
  {DESCENDING}
    [DUPLICATES ARE [FIRST] ALLOWED]
                    [LAST ]
                    [NOT  ] ]
[;SEARCH KEY IS data-base-identifier-5 [,data-base-identifier-6]...
    [USING { CALC                                  } ] DUPLICATES ARE [NOT] ALLOWED ]...
           { INDEX [NAME IS index-name-1]          }
           { data-base-procedure-1                 }
```

Graphe correspondant



SET-TYPE : désigne le concept de type de set

MODE : désigne les modes d'organisation de set. (la clause MODE IS ...)

ORDER : désigne l'ordre d'un set (la clause ORDER)

INDEX-REC(SET-TYPE, OTHERS) désigne le record index associé à un ordre de tri et défini par constructeur (la clause NAME is index-name-1)

OWNER(REC-TYPE, SET-TYPE) désigne le type owner du type de set

(SET-TYPE, MEMBER) et (MEMBER, REC-TYPE) désignent les types member du type de set

(MEMBER, TYPE) désigne la clause MANDATORY/OPTIONAL et AUTOMATIC/MANUAL

DUP-NOT désigne la clause DUPLICATES NOT ALLOWED FOR ... (pour déclarer un identifiant de set)

(DUP-NOT, ITEM) désigne un identifiant de set constitué par un nombre d'items

SORT-KEY désigne l'ordre de tri des records member d'un type donné (représenté par la relation TRI(MEMBER, SORT-KEY)), selon les valeurs d'items (représenté par la relation USING (SORT-KEY, ITEM)).c'est la clause ASCENDING/DESCENDING-KEY.

(SORT-KEY, DUP) désigne l'action à exécuter dans le cas où les valeurs de clé de tri sont identiques (l'option DUPLICATES ARE ... de la clause ASCENDING/DESCENDING KEY)

SEARCH-KEY désigne la clause SEARCH KEY (clé d'accès aux records member)

(SEARCH-KEY, DUP) désigne l'action à exécuter dans le cas où les valeurs de la clé d'accès sont identiques (l'option DUPLICATES de la clause SEARCH KEY)

INDEX-REC (SEARCH-KEY, OTHERS) désigne le record index associé à la clé d'accès aux records member et défini par constructeur

OTHERS désigne un "objet" défini par constructeur

(SEARCH-KEY, PROCED-PAR) : une procédure associée à une clé d'accès aux records member.

Contraintes d'intégrité

Les items élémentaires participant à une SORT-KEY, SEARCH-KEY ou DUP-NOT doivent être ceux d'un type member.

La SOS
DDL-CODASYL

Format 1

```
[;SET OCCURRENCE SELECTION IS THRU
 {CURRENT OF SET
 {LOCATION MODE OF OWNER
        [USING data-base-identifier-7 [,data-base-identifier-8]...
        {ALIAS FOR data-base-identifier-9 IS data-base-data-name-1}...]}].
```

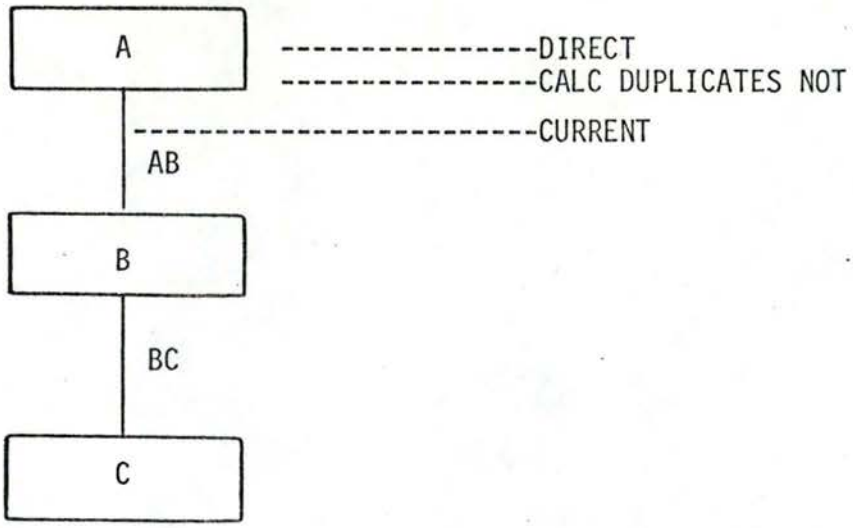
Format 2

```
[;SET OCCURRENCE SELECTION IS THRU set-name-2 USING
{CURRENT OF SET
{LOCATION MODE OF OWNER
        [ALIAS FOR data-base-identifier-10 IS data-base-data-name-2]...}
{set-name-3 {USING data-base-identifier-11 [,data-base-identifier-12]...
            {ALIAS FOR data-base-identifier-13 IS data-base-data-name-3}...}...}].
```

Pour établir un chemin d'accès dans le rapport CODASYL-71, il y a deux manières :
Le format 2 décrit explicitement les pas d'un chemin d'accès, du début jusqu'à la fin.
Le format 1 décrit implicitement les pas en se basant sur LOCATION MODE du type owner du type de set contenant cette SOS (point d'arrivée), jusqu'au point de départ.
Dans les deux cas, le point de départ doit être un set repéré sans ambiguïté c'est-à-dire :

- le courant du type du premier set de la chaine
- un type owner dont le LOCATION MODE est DIRECT
- un type owner dont le LOCATION MODE est CALC

DUPLICATES NOT ALLOWED



En examinant les deux formats, on constate que le format 1 est un cas particulier du format 2, en effet, le format 1 décrit implicitement un chemin d'accès, ce qui n'est possible qu'à condition qu'il n'y ait pas d'ambiguïté (c'est-à-dire qu'il existe un seul chemin d'accès possible, par conséquent, il n'y a pas de clause "THRU set-name")

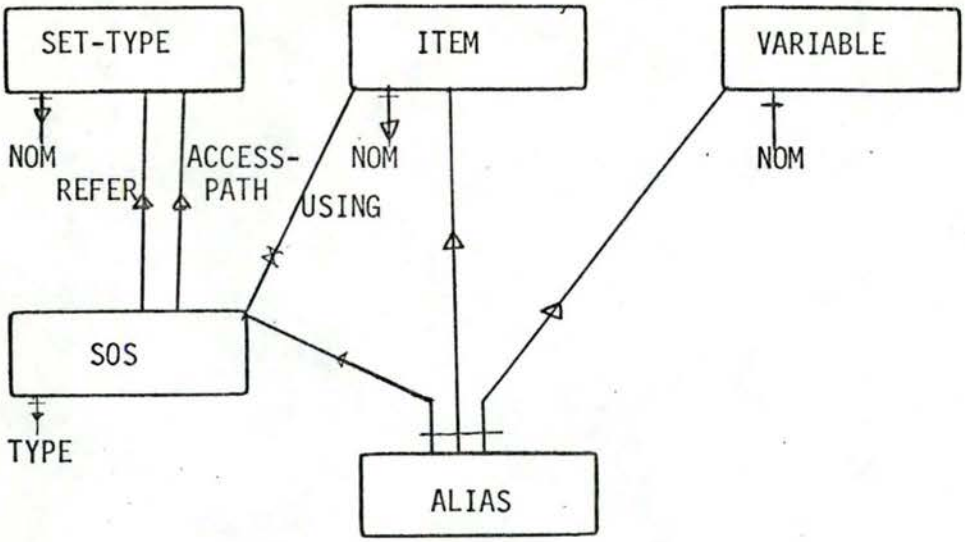
On peut donc, grouper les 2 formats dans le schéma conceptuel en utilisant les deux relations :

- ACCESS-PATH décrit des chemins d'accès
  .ACCESS-PATH doit avoir un ordre bien déterminé (par convention, nous choisissons l'ordre du format 2,du point de départ jusqu'au point d'arrivée)
  .le format 1 demande un examen de "LOCATION MODE" du type owner du type de set spécifié avant l'établissement d'un pas du chemin d'accès. Cet examen se poursuit jusqu'au point de départ (LOCATION MODE du type owner est DIRECT ou CALC DUPLICATES NOT ou VIA avec la SOS-CURRENT du type de set indiqué)
- REFER : permet de référencer un type de set.

La clause "ALIAS"

Cette clause permet d'enlever des ambiguïtés quant le repérage de sets, grâce à des identifiants globaux ou de set n'est pas satisfaisant.

Graphe correspondant (SOS)



Contrainte d'intégrité

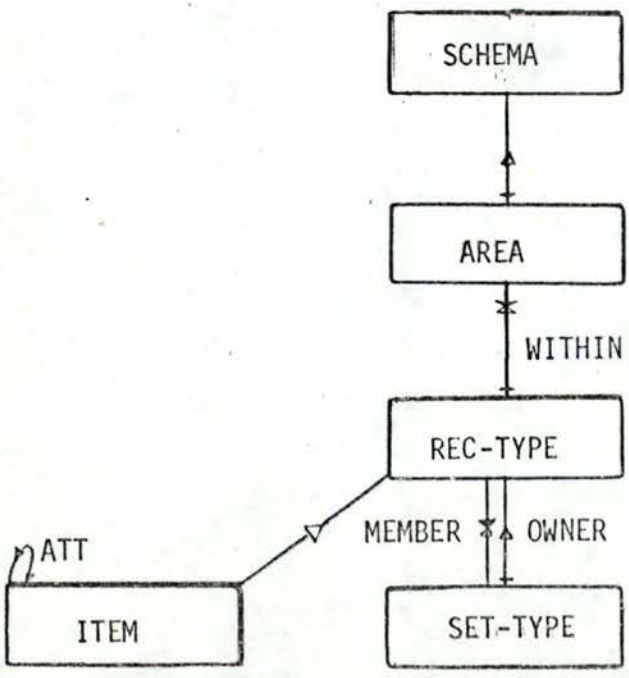
1. Les items élémentaires participant à une SOS doivent être ceux d'un type owner dans un chemin d'accès
2. La relation ACCESS-PATH doit être dans l'ordre descendant du chemin d'accès (du point de départ jusqu'au point d'arrivée)

Remarque

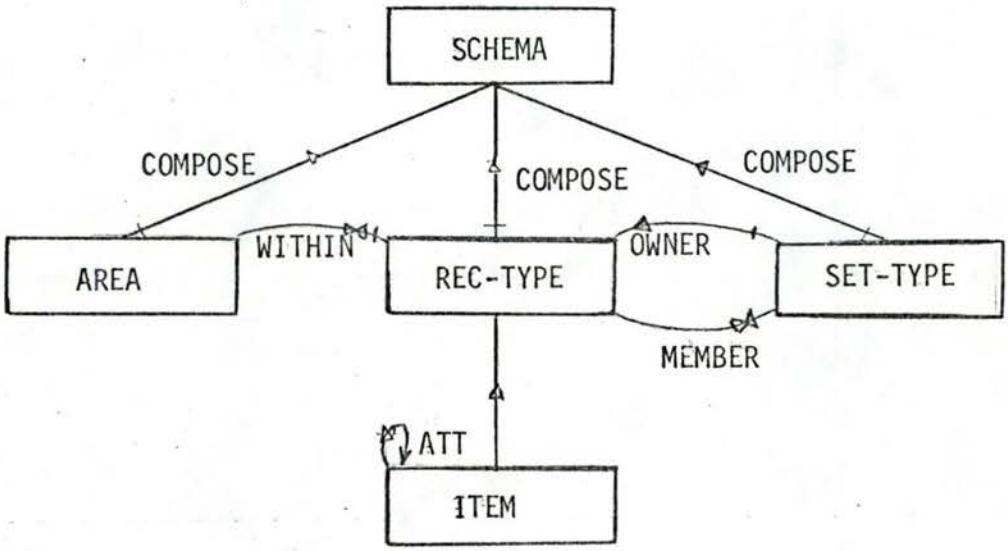
Dans un sub-schéma, on peut redéfinir la SOS.

6.2.6. Relations entre SCHEMA, AREA, REC-TYPE, ITEM, SET-TYPE

Les relations entre ces objets sans tenir compte des autres relations ou "objets" peuvent être présentées comme suit :



Mais pour mieux présenter la propriété d'appartenance à un schéma d'un type de set ou d'un type de record, nous préférons ajouter les deux relations COMPOSE (SCHEMA,SET-TYPE) et COMPOSE (SCHEMA, REC-TYPE)



Et nous ajoutons 2 contraintes d'intégrités pour exprimer les redondances :

- la relation COMPOSE(SCHEMA, REC-TYPE) est constituée à partir de la relation COMPOSE(SCHEMA,AREA) et celle de WITHIN(AREA,REC-TYPE)
- la relation COMPOSE(SCHEMA,SET-TYPE) est constituée à partir de 3 relations COMPOSE(SCHEMA,AREA), WITHIN(AREA,REC-TYPE) et OWNER(REC-TYPE,SET-TYPE).

6.2.7. Sub-schéma

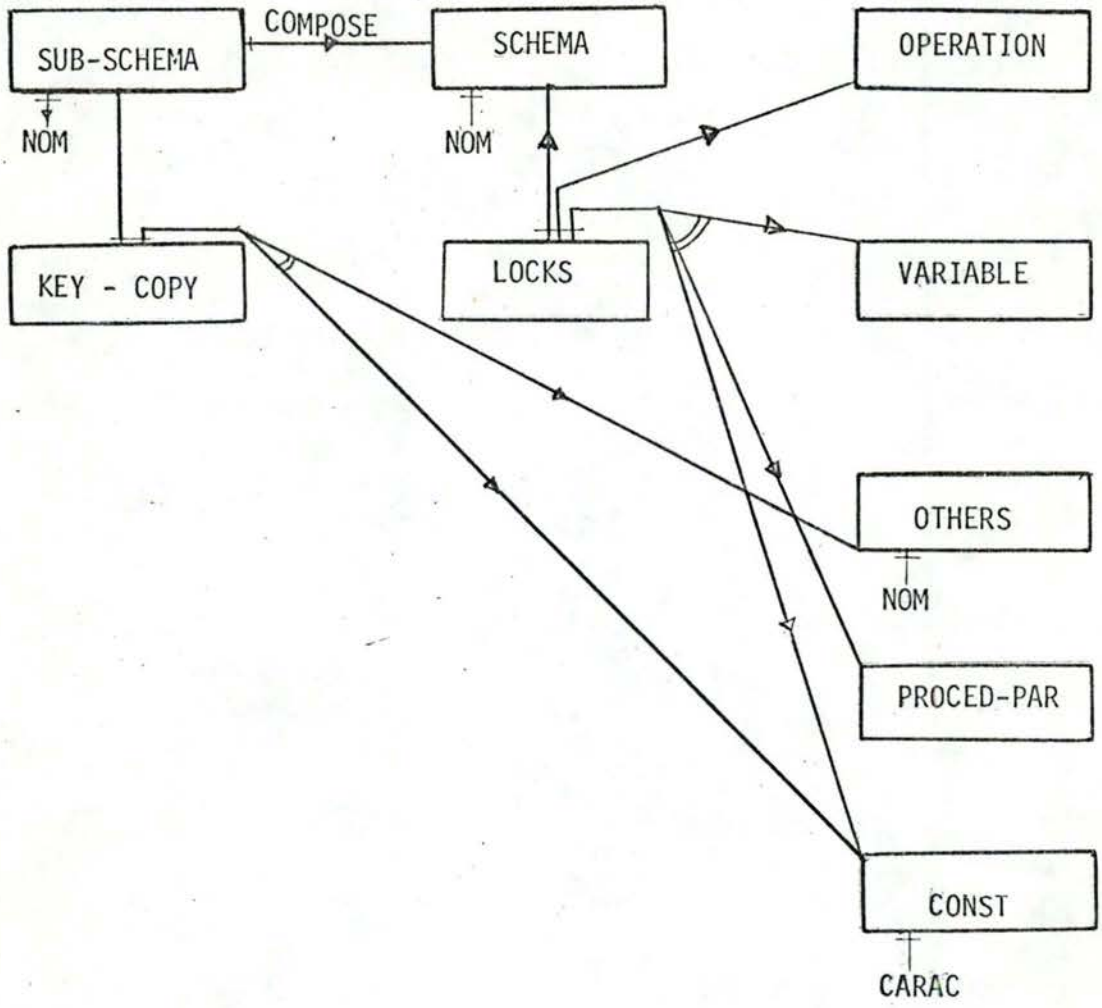
Un sub-schéma est une description partielle d'une base de données, il est connu par les programmes d'application.
On le désigne par un nom.

DDL-CODASYL

SUB-SCHEMA NAME IS sub-schema-name OF SCHEMA NAME schema-name

$$\left[\text{PRIVACY KEY FOR COPY IS } \left\{\begin{matrix}\text{literal-5}\\ \text{implementor-name-1}\end{matrix}\right\}\right] .$$

Graphe correspondant



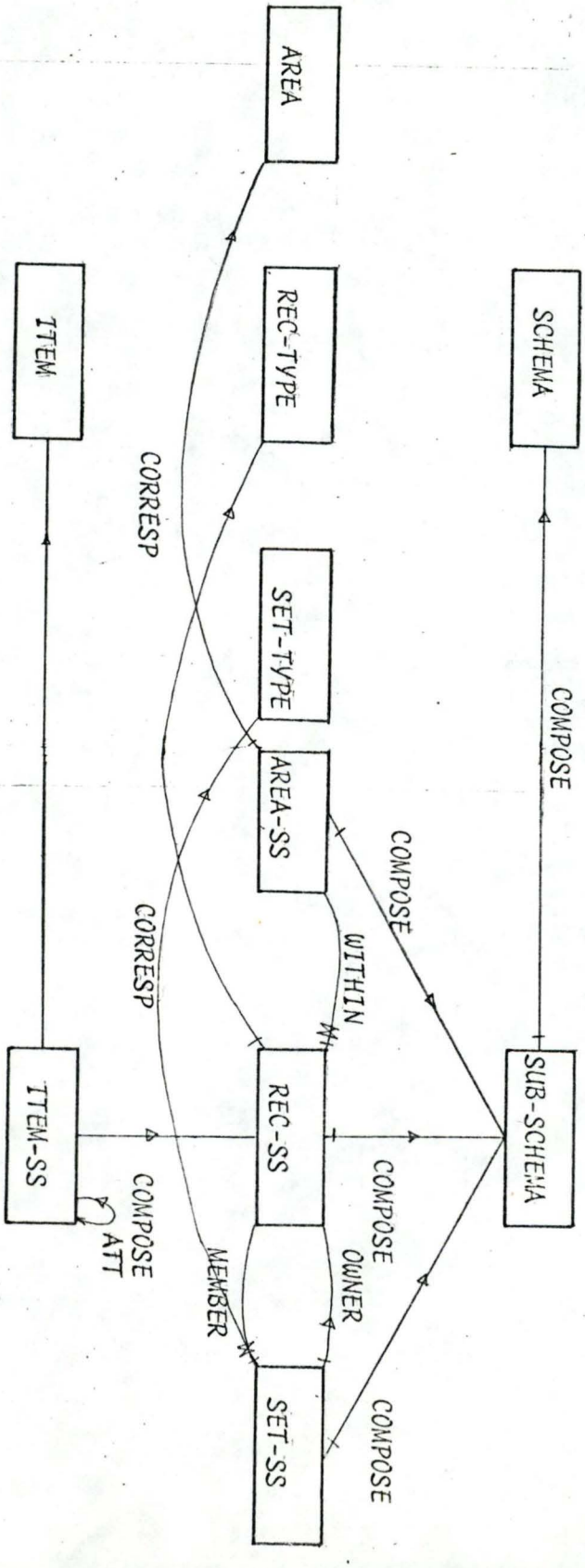
Le SGBD-CODASYL s'effectue éventuellemcnt un contrôle à partir de la clause KEY FOR COPY et le "LOCKS" correspondant.

Contrainte d'intégrité

1. Le nom d'un sub-schéma doit être unique dans un schéma
2. La clé literal-5 doit être égale au verrou correspondant du schéma.

6.2.8. Relations entre les objets de schéma et ceux de sub-schéma.

Si l'on ne considère que les objets : SEHCMA, AREA, REC-TYPE, ITEM, SET-TYPE et ceux du sub-schéma, leurs relations peuvent être présentées comme suit :



CORRESP(AREA-SS,AREA) représente la relation entre une aréa déclarée dans un sub-schéma et celle déclarée dans le schéma correspondant.

Si l'on conserve les deux relations COMPOSE (SUB-SCHEMA, REC-SS) et COMPOSE (SUB-SCHEMA, SET-SS) pour référer le REC-SS et le SET-SS au sub-schéma spécifié, les trois relations

- WITHIN(REC-SS,AREA-SS)
- OWNER(SET-SS,REC-SS)
- MEMBER(SET-SS, REC-SS)

sont redondantes.

En effet, un sub-schéma est une description partielle d'une BD, en se basant sur le schéma correspondant, les aréas, les types de record, les types de set du sub-schéma considéré doivent être déclarés dans le schéma; ce qui entraîne que les réalisations de l'une des trois relations précédentes sont un sous-ensemble des réalisations de la relation du même type du schéma (les trois relations WITHIN(REC-TYPE, AREA), OWNER(SET-TYPE,REC-TYPE) et MEMBER(SET-TYPE,REC-TYPE))

Contraintes d'intégrité

Les réalisations de la relation WITHIN(REC-SS, AREA-SS) (respectivement OWNER (SET-SS,REC-SS), MEMBER(SET-SS, REC-SS)) doivent être un sous-ensemble de celles de WITHIN(REC-TYPE,AREA)(respectivement OWNER(SET-TYPE,REC-TYPE) et MEMBER(SET-TYPE,REC-TYPE))

6.2.9. Renaming section

Les noms des types de données d'un sub-schéma associé peuvent être différents à ceux du schéma correspondant.

DDL-CODASYL

RENAMING SECTION

```
[AREA NAME area-name-1 IN SCHEMA IS CHANGED TO area-name-2
      [,area-name-3 TO area-name-4]... ]...

[RECORD NAME record-name-1 IN SCHEMA IS CHANGED TO record-name-2
      [,record-name-3 TO record-name-4]... ]...
```

[DATA NAME data-base-identifier-1 IN SCHEMA IS CHANGED TO data-base-data-name-1
[,data-base-identifier-2 TO data-base-data-name-2]...] ...

[SET NAME set-name-1 IN SCHEMA IS CHANGED TO set-name-2
[,set-name-3 TO set-name-4]...] ...

[IMPLEMENTOR NAME implementor-name-1 IN SCHEMA IS CHANGED TO implementor-name-2
[,implementor-name-3 TO implementor-name-4]...] ... .

Ce qui entraîne qu'il existe éventuellement un nom associé à un objet d'un sub-schéma.

6.2.10. Aréa d'un sub-schéma

Une aréa d'un sub-schéma (AREA-SS) est une subdivision d'une BD, auquel les utilisateurs peuvent accéder via ce sub-schéma.

DDL-CODASYL

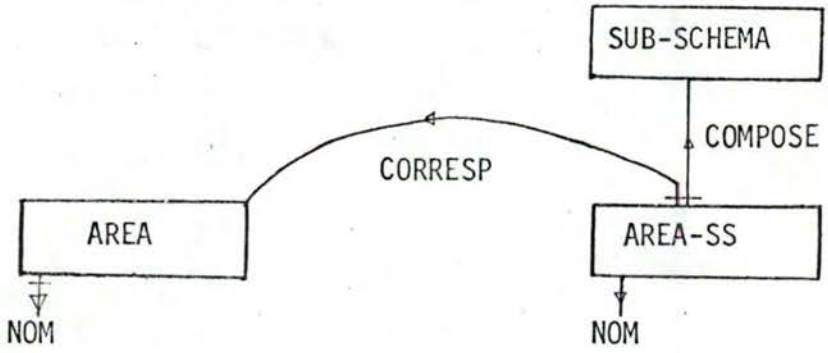
Format 1

COPY area-name-1 [,area-name-2] ...

Format 2

COPY ALL AREAS .

Graphe correspondant



Contrainte d'intégrité

Le nom d'aréa dans un sub-schéma doit être unique.

6.2.11. Type de record et items associés d'un sub-schéma

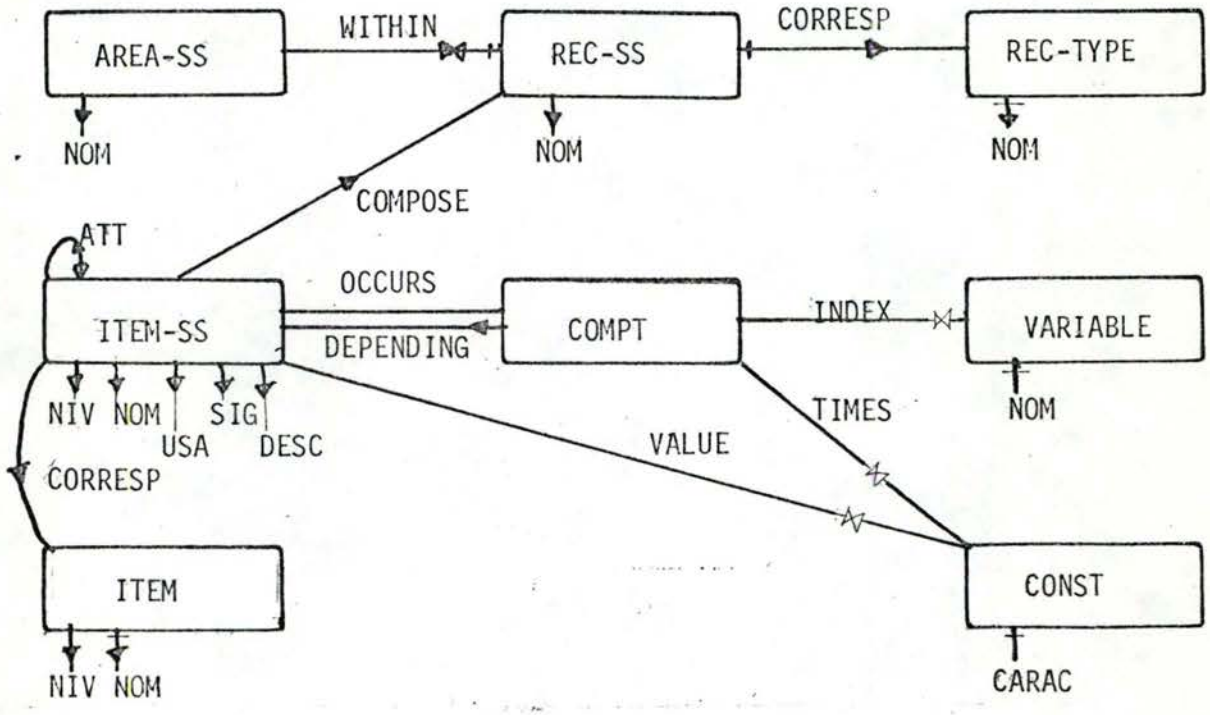
Un type de reocrd d'un sub-schéma est un ensemble de types de données qui définissent un même type d'entité, un même type d'individu, il est associé à celui du schéma correspondant et connu par les programmes utilisant.

Un item associé à un type de record d'un sub-schéma est celui connu par utilisateur (ITEM-SS).

DDL-CODASYL

```
01 record-name-1
[;WITHIN area-name-1 [,area-name-2] ...]

level-number   data-base-data-name-1

[; { PICTURE } IS character-string ]
   { PIC     }

[; [USAGE IS] { COMPUTATIONAL   } ]
              { COMP            }
              { COMPUTATIONAL-n }
              { COMP-n          }
              { DISPLAY         }
              { DATABASE-KEY    }

[; [SIGN IS] { LEADING  } [SEPARATE CHARACTER] ]
             { TRAILING }

[;OCCURS integer-2 TIMES
   [INDEXED BY index-name-1 [,index-name-2] ...]
;OCCURS integer-1 TO integer-2 TIMES
   DEPENDING ON data-base-data-name-3
   [INDEXED BY index-name-1 [,index-name-2] ...] ]
```

```
88 condition-name; { VALUE IS   } literal-1 [ { THRU    } literal-2 ]
                   { VALUES ARE }           { THROUGH }

[ ,literal-3 [ { THRU    } literal-4 ] ] ...
               { THROUGH }
```



(ITEM-SS,NIV) : désigne un niveau hiérarchique

(ITEM-SS, NOM) : désigne le nom d'un item

(ITEM-SS, USA) : désigne les modes de représentation des valeurs d'items (La clause USAGE)

(ITEM-SS, SIG) : désigne la position et le mode de représentation du signe +/- d'un item numérique, (il est associé à DESC)(la clause SIGN)

(ITEM-SS, DESC) : désigne la description d'items (la clause PICTURE)

INDEX(COMPT, VARIABLE) : désigne un ou plusieurs index associé à un vecteur ou groupe répétitf (l'option INDEX de la clause OCCURS)

VALUE(ITEM-SS, CONST) : désigne la clause "condition-name"

Contraintes d'intégrité

1. La qualification des noms d'items doit être unique relativement à un sub-schéma
2. TIMES(COMPT, CONST) ≡ 0-2, 0-∞
3. ATT(ITEM-SUB, ITEM-SUB) doit être dans l'ordre de déclaration
4. Un item (élémentaire ou composé) doit être associé à son type de record par les 2 relations COMPOSE(REC-TYPE, ITEM-SS) et ATT(ITEM-SS, ITEM-SS) Ces deux relations forment un arbre hiérarchique.

6.2.12. Type de set d'un sub-schéma

Un type de set d'un sub-schéma est une relation entre type owner et des types members appartenant à ce sub-schéma.

DDL-CODASYL

SET SECTION.

Format 1

COPY set-name-1 [,set-name-2] ... .

Format 2

COPY ALL SETS.

Format 3

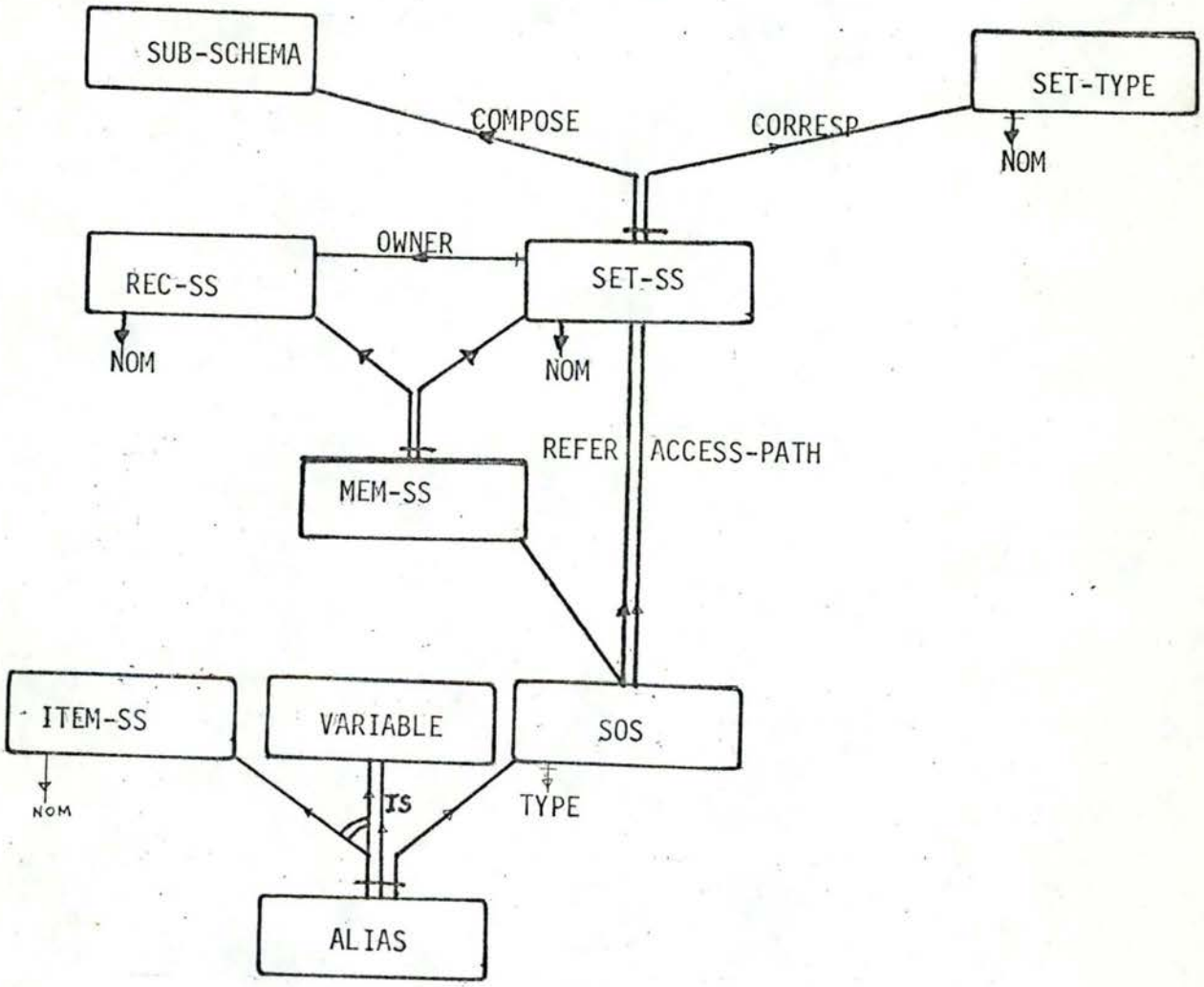
COPY set-name-3

SET OCCURRENCE SELECTION FOR MEMBER record-name-1 IS THRU set-name-4 USING

{CURRENT SET
{LOCATION MODE OF OWNER [ALIAS FOR data-base-identifier-1 IS
data-base-data-name-1] ...}

{set-name-5 {USING data-base-identifier-2 [,data-base-identifier-3]...
{(ALIAS FOR data-base-identifier-4 IS data-base-data-name-2)...}}...

Graphe correspondant



Dans un sub-schéma, on peut changer de critère de sélection de set déclaré dans le schéma correspondant ce qui exprime l'existence des relations :

ACCESS-PATH(SET-SS, SOS)

REFER(SET-SS, SOS)

(ITEM-SS, ALIAS), (VARIBALE, ALIAS)

IS(VARIABLE, ALIAS)

(SOS, ALIAS)

La relation (MEM-SS, SOS) permet de préciser le type member du type de set du sub-schéma (ce type de set subit une modification de critère de sélection).

Contrainte d'intégrité

1. Le nom d'un type de set doit être unique dans ce sub-schéma
2. Un type de set d'un sub-schéma doit avoir son type owner qui appartient à ce sub-schéma.

6.2.13. Contrôle d'autorisation

Un contrôle d'autorisation est une relation entre un objet, une opération, et un "user".

Les domaines de valeurs sont présentés dans TABLE 2.1.

DDL-CODASYL

a) au niveau du schéma

```
[ ;PRIVACY LOCK [ FOR { LOCKS | DISPLAY | COPY | ALTER } ] IS { literal-1 | lock-name-1 | PROCEDURE data-base-procedure-1 }
    [ OR { literal-2 | lock-name-2 | PROCEDURE data-base-procedure-2 } ] ... ] ... .
```

b) au niveau d'aréa

```
[ PRIVACY LOCK [ FOR { [EXCLUSIVE | PROTECTED] RETRIEVAL
                    | [EXCLUSIVE | PROTECTED] UPDATE
                    | support-function-1 [, support-function-2] ... } IS
{ literal-1 | lock-name-1 | PROCEDURE data-base-procedure-1 }
[ OR { literal-3 | lock-name-2 | PROCEDURE data-base-procedure-2 } ] ... ] ... .
```

c) au niveau du type de record

```
[;PRIVACY LOCK [FOR [ INSERT  ]] IS { PROCEDURE data-base-procedure-1 }
               [    | REMOVE  |]    { literal-1                       }
               [    | STORE   |]    { lock-name-1                     }
               [    | DELETE  |]
               [    | DELETE ONLY |]
               [    | DELETE SELECTIVE |]
               [    | DELETE ALL |]
               [    | GET     |]
               [    | MODIFY  |]
               [    | FIND    |]

   [ OR { PROCEDURE data-base-procedure-2 } ] ... ] ...
        { literal-3                       }
        { lock-name-2                     }
```

d) au niveau d'item

```
[; PRIVACY LOCK [FOR [ STORE ]] IS { PROCEDURE data-base-procedure-5 }
                [    | GET   |]    { literal-3                       }
                [    | MODIFY|]    { lock-name-1                     }

   [ OR { PROCEDURE data-base-procedure-6 } ] ... ] ...
        { literal-4                       }
        { lock-name-2                     }
```

e) au niveau du type de set

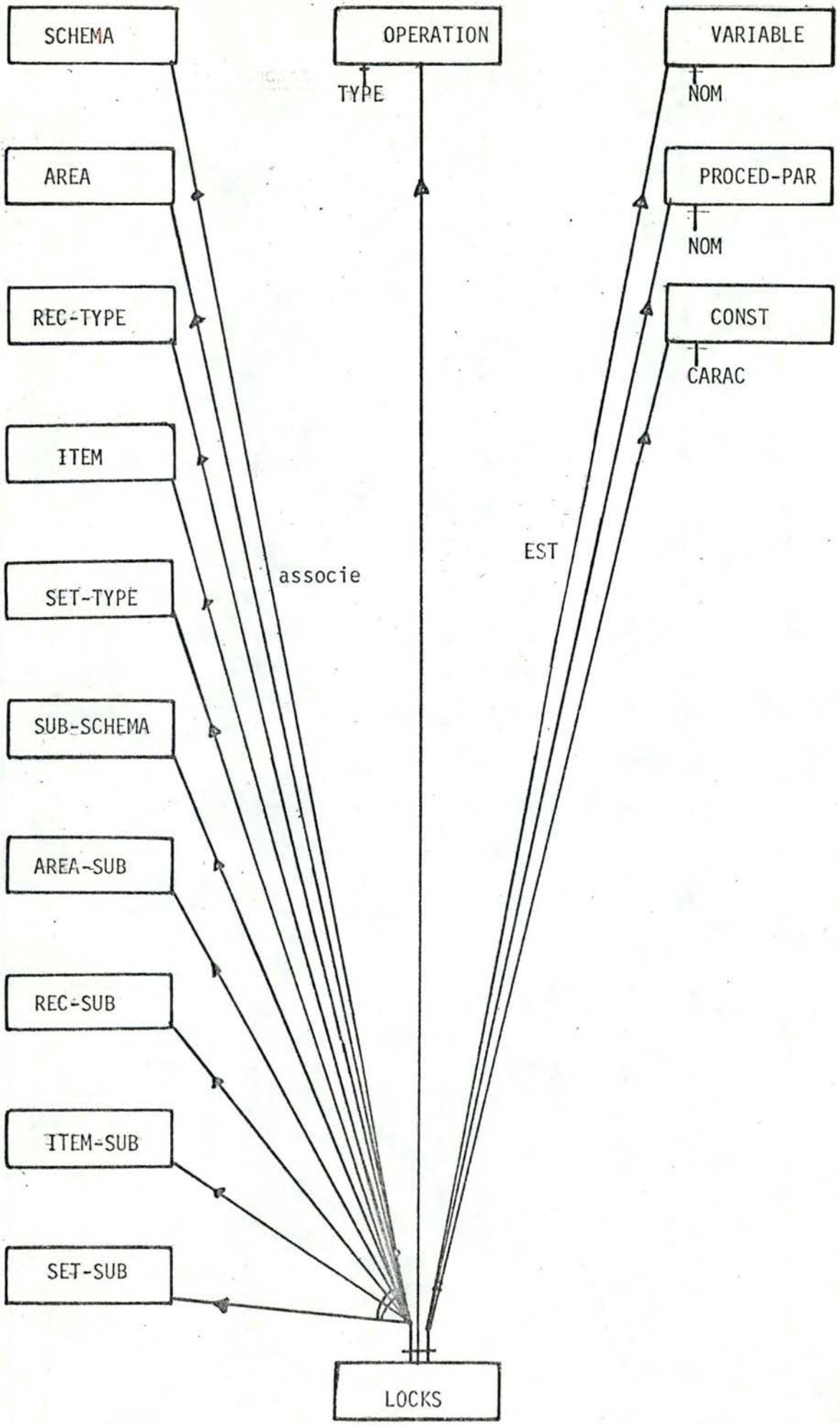
```
[PRIVACY LOCK [FOR [ ORDER  ]] IS { literal-1                       }
              [    | FIND   |]    { lock-name-1                     }
              [    | REMOVE |]    { PROCEDURE data-base-procedure-1 }
              [    | INSERT |]

   [ OR { literal-3                       } ] ... ] ...
        { lock-name-2                     }
        { PROCEDURE data-base-procedure-2 }
```

f) au niveau du sub-schéma

.....

```
[ PRIVACY LOCK [ FOR [ LOCKS   ] ] ] IS { literal-1          }
               [     [ DISPLAY ] ]       { lock-name-1       }
               [     [ COMPILE ] ]       { PROCEDURE data-base-procedure-1 }
               [     [ ALTER   ] ]

    [ [ OR { literal-3          } ] ... ] ...
           { lock-name-2        }
           { PROCEDURE data-base-procedure-2 }
```

Graphe correspondant



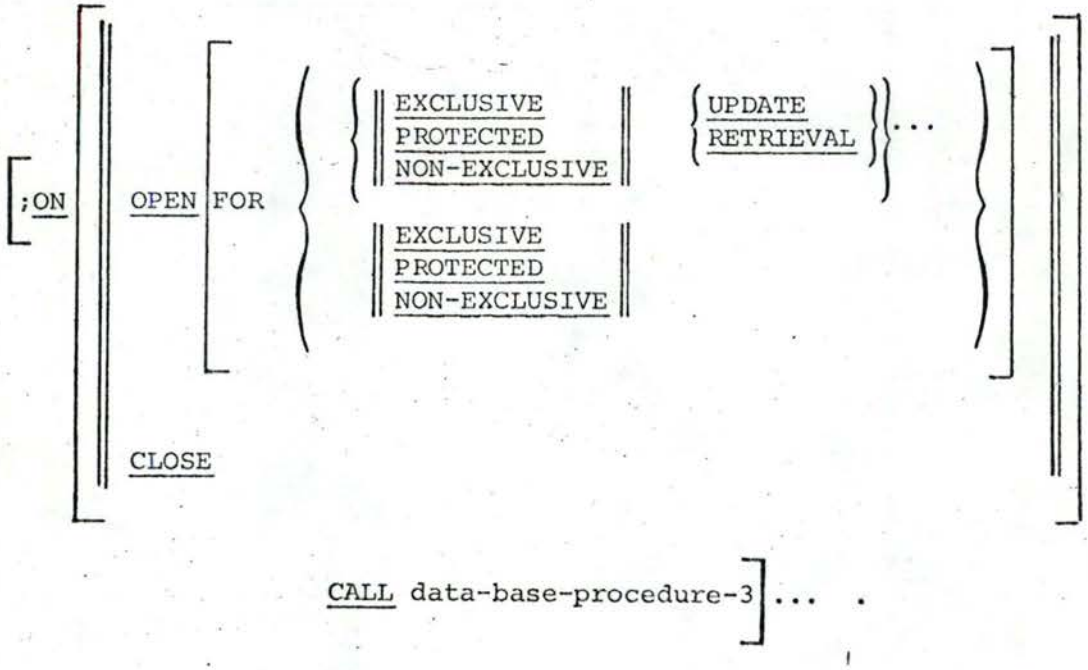
Le contrôle d'autorisation associé à un objet (la relation ASSOCIE(LOCKS, objet)) pour une opération (la relation POUR (LOCKS, OPERATION)) est soit une constante (la relation EST(LOCKS,CONST)) soit une variable contenant une valeur (la relation EST(LOCKS, VARIABLE)
soit une procédure de contrôle (la relation EST(LOCKS, PROCEDURE-PAR)
d'où : objet ≡ SCHEMA,AREA,REC-TYPE ...

6.2.14. Les traitements (procédures) particuliers associés aux opérations

Il s'agit des procédures autres que celles déjà citées jusqu'à présent. Nous représentons donc la clause "ON".

DDL-CODASYL

a) au niveau d'aréa



b) au niveau du type de record

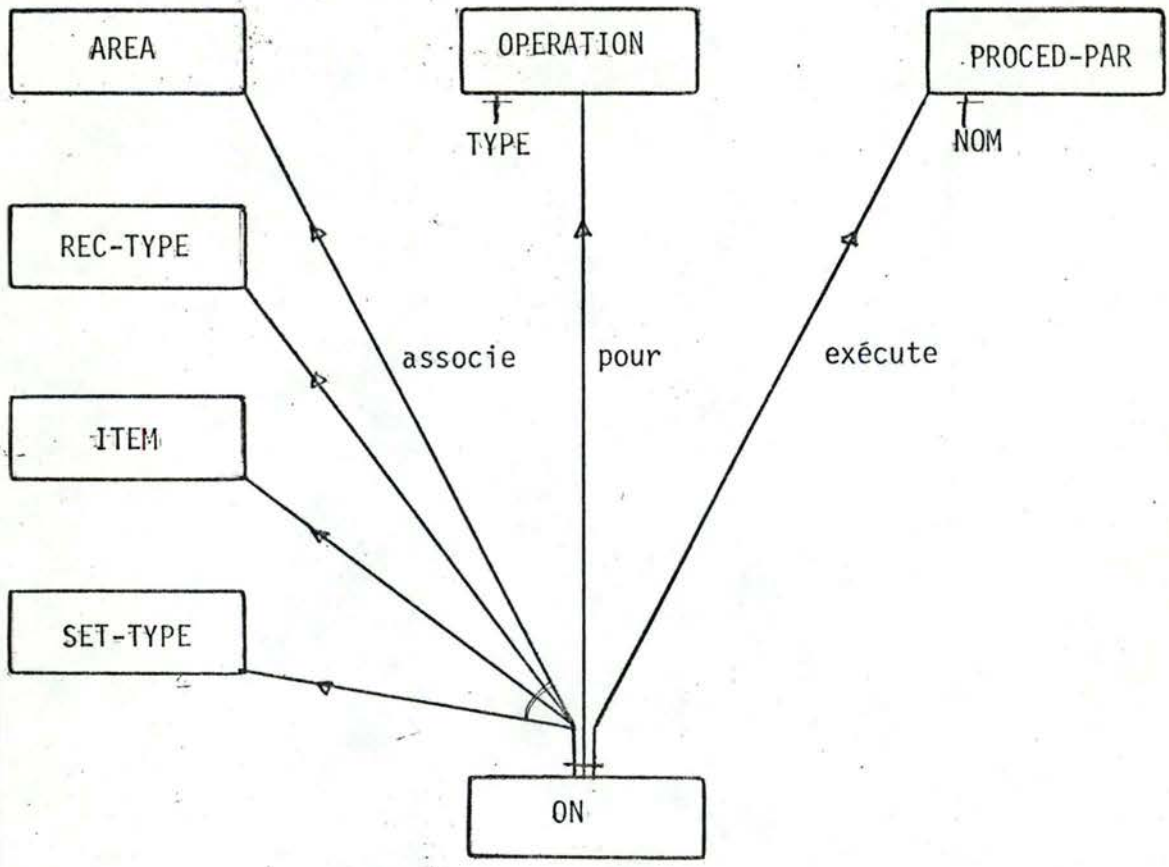
```
[;ON [ INSERT                                  ] CALL data-base-procedure-2 ]...
     [ REMOVE                                  ]
     [ STORE                                   ]
     [ DELETE                                  ]
     [ DELETE ONLY                             ]
     [ DELETE SELECTIVE                        ]
     [ DELETE ALL                              ]
     [ MODIFY                                  ]
     [ FIND                                    ]
     [ GET                                     ]
```

c) au niveau d'item

```
[; ON [ STORE ] CALL data-base-procedure-3
      [ GET   ]
      [ MODIFY ]
        [USING data-base-identifier-6 [,data-base-identifier-7]... ] ]...
```

d) au niveau du type de set

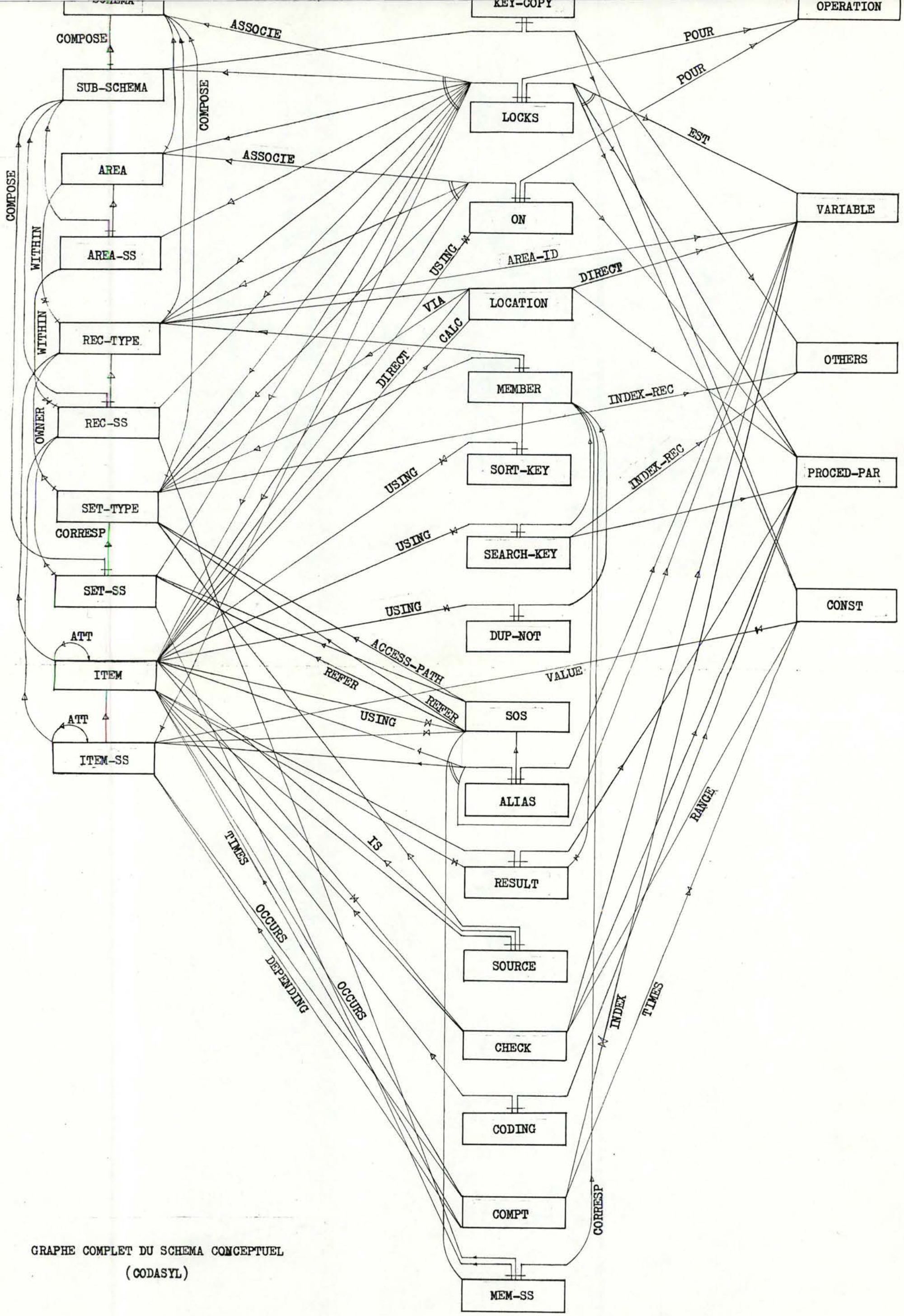
```
[;ON [ ORDER  ] CALL data-base-procedure-1 ]...
     [ INSERT ]
     [ REMOVE ]
```



Une procédure est exécutée (la relation EXECUTE(PROCED-PAR,ON)) avant ou après une opération. (la relation POUR(OPERATION,ON)) associée à un objet (la relation ASSOCIE(objet,ON)). d'où : objet ≡ AREA,REC-TYPE,ITEM, SET-TYPE.

6.3. *Graphe complet du schéma conceptuel*

Le dispositif de la figure ne nous permet pas de présenter certaines contraintes d'intégrité exprimées dans les graphes partiels.



GRAPHE COMPLET DU SCHEMA CONCEPTUEL

(CODASYL)

6.4. Remarque

Nous avons représenté les propriétés et les règles associées aux types de données du modèle CODASYL par un schéma conceptuel comportant un ensemble de contraintes d'intégrité.

Mais nous n'avons pas examiné toutes les contraintes d'intégrité des types de données CODASYL. D'une part, le rapport CODASYL-71 laisse aux constructeurs différents choix, d'autre part, notre travail est basé sur des schémas CODASYL supposés corrects, tant au point de vue syntaxique que celui sémantique.

°
° °

Dans ce chapitre, nous tentons de représenter des types de données DBMS-20 par un schéma conceptuel, leurs propriétés sont moins nombreuses que celles du CODASYL-71, en particulier toutes les procédures de gestion sont standardisées. Les significations des types de données DBMS-20 et les contraintes d'intégrité sont en général les mêmes que ceux du CODASYL-71, en conséquence, nous ne prenons que le DDL-DBMS-20 et nous le présentons par les graphes.

Les pseudonymes sont destinés aux programmes FORTRAN, nous ne les retenons pas dans les graphes.

7.1. Le schéma DBMS-20

1. DDL-DMBS-20

a) Schéma

SCHEMA NAME IS schéma-name

b) Aréa

AREA NAME IS area-name-1
[AREA IS TEMPORARY]
[[PRIVACY LOCK [FOR [EXCLUSIVE / PROTECTED] {UPDATE / RETRIEVAL}] IS lock-1]].

c) Types de record

RECORD NAME IS record-name-1

LOCATION MODE IS { DIRECT identifier-1 [%pseudonym-1] / CALC USING data-name-1 [data-name-2] ... [DUPLICATES ARE [NOT] ALLOWED] / VIA set-name-1 }

WITHIN area-name-1 [area-name-2 ... AREA-ID IS identifier-2 [%pseudonym-2]].

02 data-name-3 [%pseudonym-3] { PICTURE ... / SIZE ... / TYPE ... } [OCCURS ...].

d) items associés

Format 1

```
02 data-name-3 [%pseudonym-3] { PICTURE } IS picture-string
                              { PIC     }

[ USAGE IS  DISPLAY   ]
[           DISPLAY-6 ]  [OCCURS integer-1 TIMES].
[           DISPLAY-7 ]
[           DISPLAY-9 ]
```

Format 2

```
02 data-name-3 [%pseudonym-3] SIZE IS integer-2 { WORDS                          }
                                                { USAGE { DISPLAY   }            }
                                                {       { DISPLAY-6 }            }
                                                {       { DISPLAY-7 }            }
                                                {       { DISPLAY-9 }            }

[OCCURS integer-1 TIMES].
```

Format 3

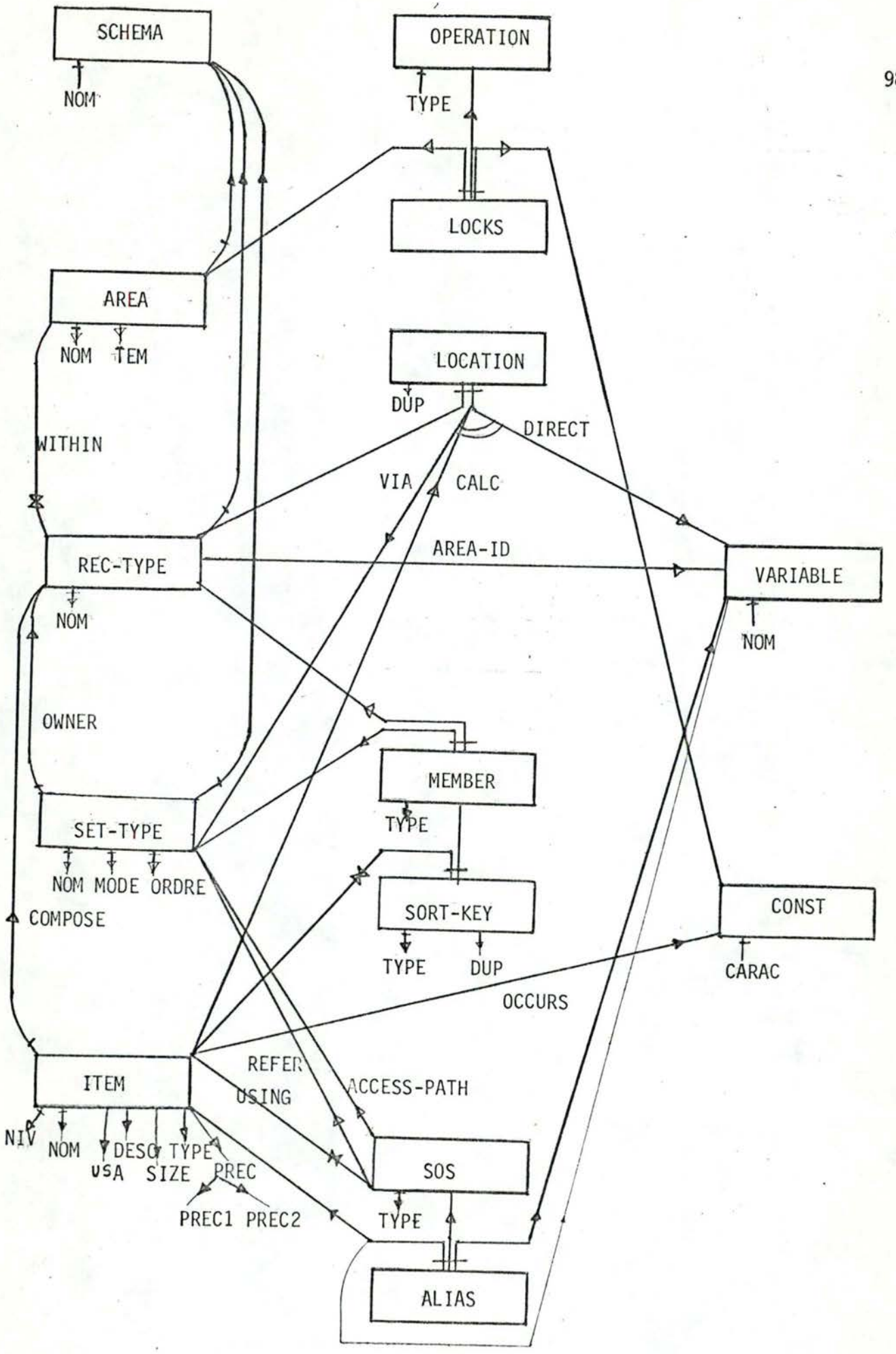
```
02 data-name-3 [%pseudonym-3]  TYPE IS { [FLOAT  ] [ { DECIMAL } ] [ REAL    ] }
                                       { [FIXED  ] [ { DEC     } ] [ COMPLEX ] }
                                       { DBKEY                                 }

[integer-3] [, integer-4] }  [OCCURS integer-1 TIMES].
                          }
```

e) Types de set

```
SET NAME IS set-name-1

MODE IS CHAIN [LINKED TO PRIOR]

                { ALWAYS { FIRST }                                                  }
                {        { LAST  }                                                  }
                {        { NEXT  }                                                  }
                {        { PRIOR }                                                  }
ORDER IS        {                                                                   }
                { SORTED [ WITHIN RECORD-NAME                                     ] }
                {        [ BY DATABASE-KEY                                        ] }
                {        [                                                        ] }
                {        [ DUPLICATES ARE [ FIRST ]           ALLOWED             ] }
                {        [                [ LAST  ]                               ] }
                {        [                [ NOT   ]                               ] }

OWNER IS { record-name-1 } [.]
         { SYSTEM        }
```

```
MEMBER IS record-name-1   {MANDATORY}  {AUTOMATIC}
                          {OPTIONAL  }  {MANUAL   }
    [LINKED TO OWNER]
[{ASCENDING                }
 {DESCENDING               }   KEY IS data-name-1 [data-name-2] ...
 {ASCENDING  RANGE         }
 {DESCENDING RANGE         }]
    [DUPLICATES ARE [FIRST]  ALLOWED]
                    [LAST ]
                    [NOT  ]
    [SET OCCURRENCE SELECTION IS THRU  {CURRENT OF SET          }
                                       {LOCATION MODE OF OWNER  }
[{USING data-name-2 [data-name-3]. . .                          }]}] [.]
 {ALIAS FOR data-name-4 IS identifier-1 [%pseudonym-1]. . .     }
```

2. Graphe correspondant


SCHEMA
NOM
OPERATION
TYPE
LOCKS
AREA
NOM
TEM
LOCATION
DUP
DIRECT
WITHIN
VIA
CALC
REC-TYPE
AREA-ID
VARIABLE
NOM
NOM
OWNER
MEMBER
TYPE
SET-TYPE
NOM
MODE
ORDRE
COMPOSE
SORT-KEY
TYPE
DUP
CONST
CARAC
OCCURS
ITEM
REFER
USING
ACCESS-PATH
NIV
NOM
DESC
TYPE
USA
SIZE
PREC
PREC1
PREC2
SOS
TYPE
ALIAS

(ITEM,USA) représente la clause USAGE
(ITEM,DESC) représente la clause PICTURE
(ITEM,Size) représente la clause SIZE (item composé)
(ITEM,Type) représente la clause TYPE (item numérique interne)
(ITEM,PREC) représente l'option "integer-1", "integer-2" de la clause TYPE

Contrainte d'intégrité

1. Les items Desc, Size, Type sont mutuellement exclusifs
2. L'item Préc est associé à l'item Type.

Remarque

Le DDL-DBMS-20 n'admet qu'au niveau du schéma les items associés de niveau 02 (la description des attributs d'item composé est décrite au niveau du sub-schéma), ce qui justifie que :

- Il n'y a pas de relation ATT(ITEM,ITEM)
- la relation COMPOSE(REC-TYPE, ITEM) est forte du côté ITEM.

7.2. *Les sub-schémas DBMS-20*

1. DDL-DBMS-20

   a) Sub-schéma

      SUB-SCHEMA NAME IS sub-schema-name
      [PRIVACY LOCK is lock-1]

   b) Aréa section

      Format 1
      AREA SECTION.
           COPY area-name-1 [area-name-2] ...
      Format 2
      AREA SECTION.
           COPY ALL AREAS.
      Format 3
      AREA SECTION.
           COPY TEMPORARY area-name-3[area-name-4] ...

c) Record section

Format 1

RECORD SECTION

01 record-name-1.
01 record-name-2.
[02 data-name-1.
[COPY sub-schéma-name TEXT.]]
[COPY OTHERS.]

Format 2

RECORD SECTION.
COPY ALL RECORDS.

d) Set section

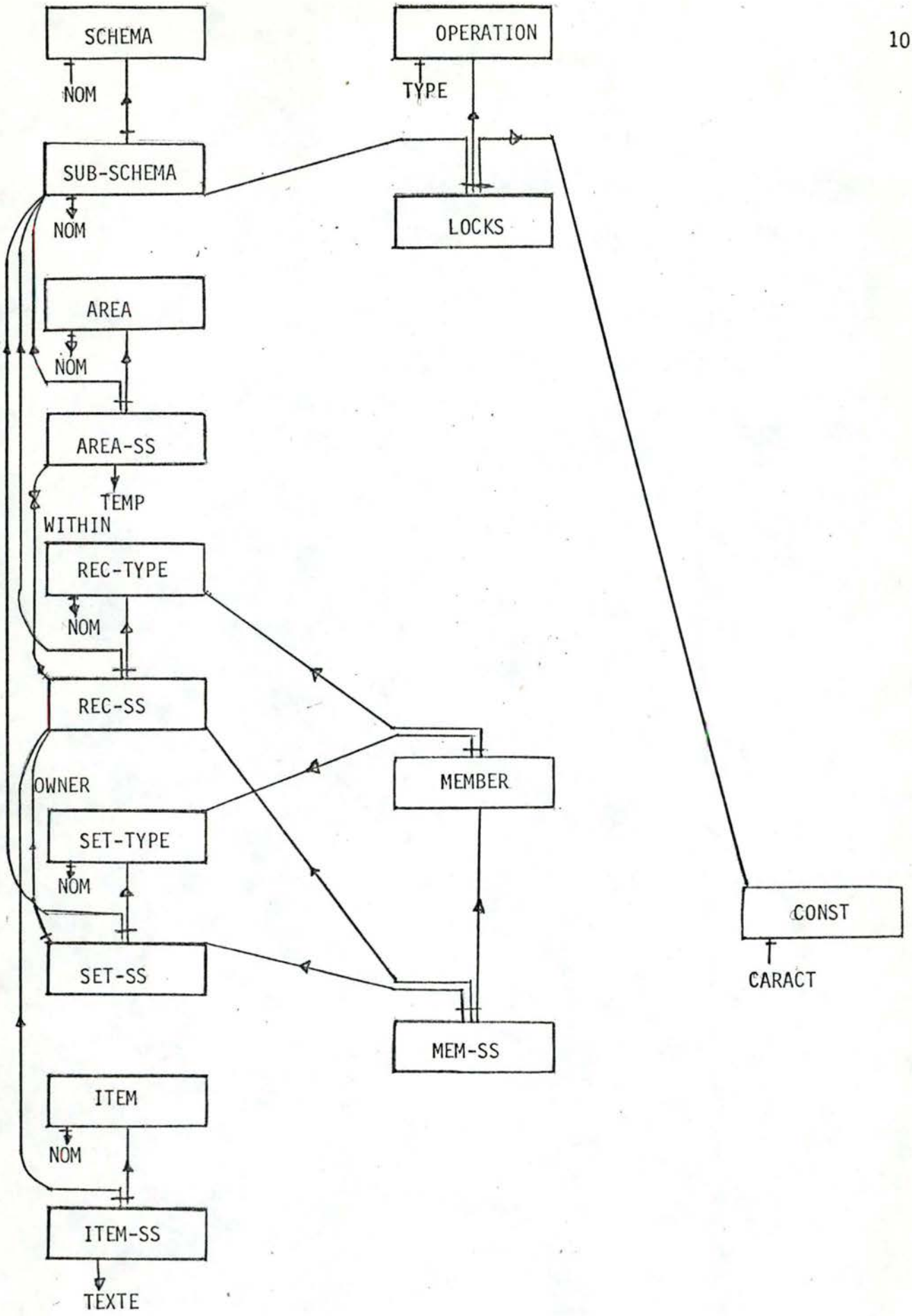
Format 1

SET SECTION.
    COPY set-name-1 [set-name-2] ...

Format 2

SET SECTION.
    COPY ALL SETS.

2. Graphe correspondant

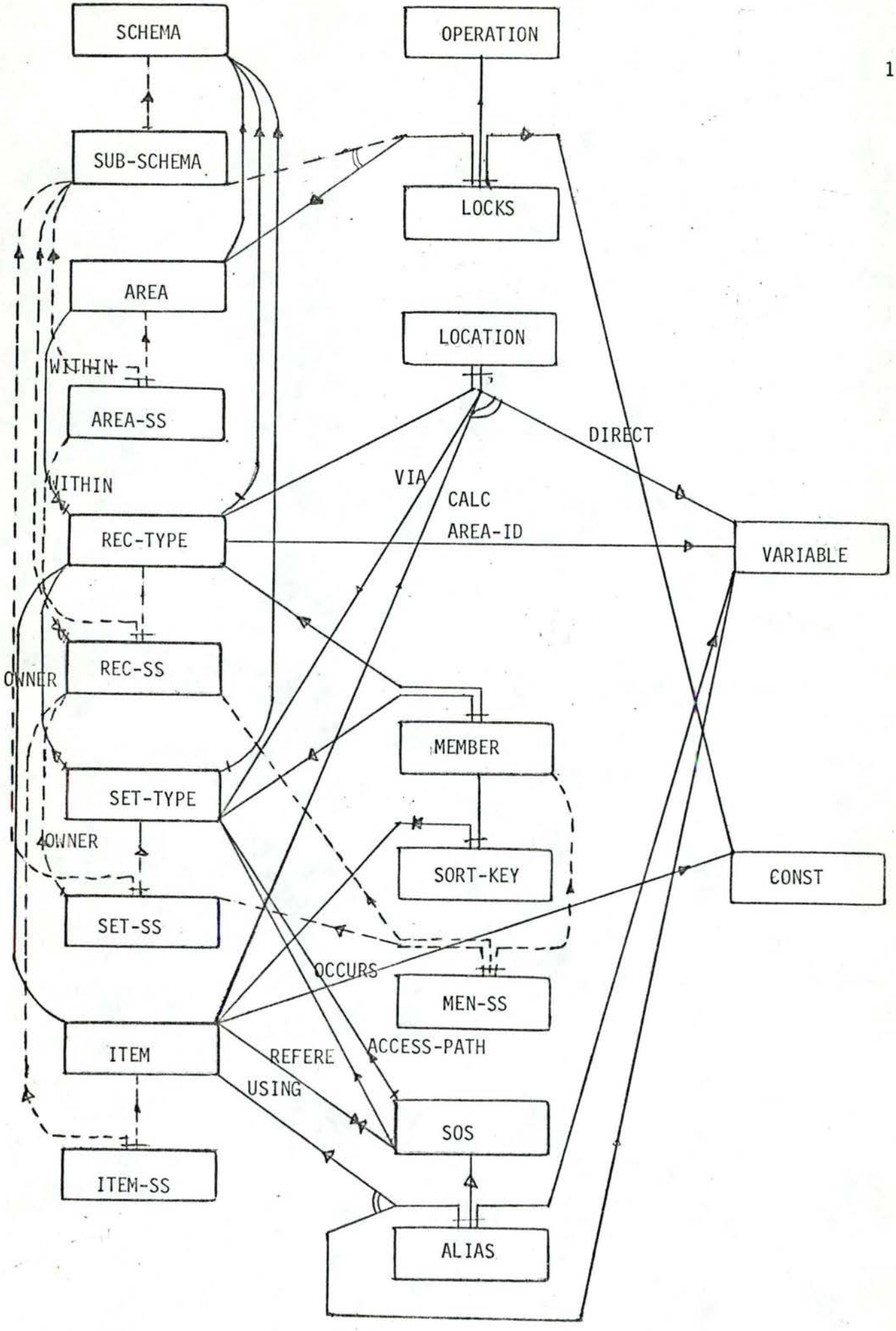


Remarque :

Le SGBD-DBMS-20 considère la description d'item composé comme un texte (ici c'est un texte COBOL). Ce texte est enregistré comme tel et est à la charge du compilateur COBOL. Ce qui nous permet de ne pas présenter la relation ATT(ITEM-SS, ITEM-SS) et la relation COMPOSE (REC-SS, ITEM-SS) est forte du côté ITEM-SS.

7.3. *Graphe complet du schéma conceptuel (DBMS-20)*

(Page suivante)


SCHEMA
OPERATION
SUB-SCHEMA
LOCKS
AREA
LOCATION
WITHIN
AREA-SS
DIRECT
VIA
CALC
AREA-ID
WITHIN
REC-TYPE
VARIABLE
REC-SS
OWNER
MEMBER
SET-TYPE
OWNER
SORT-KEY
CONST
SET-SS
OCCURS
MEN-SS
ITEM
REFERE
ACCESS-PATH
USING
SOS
ITEM-SS
ALIAS

4ème PARTIE :

REALISATION DU SCHEMA

CHAPITRE VIII : GRAPHES D'ACCES

Nous allons définir les relations d'accès indispensables au générateur de listings DDL du modèle DBMS-20 simplifié et au générateur des tables destinés au compilateur DML-MCS ainsi que de nouveaux items nécessaires à l'implémentation du schéma, en nous basant sur les deux algorithmes réalisant ces deux générateurs (ces algorithmes sont présentés au chapitre X).

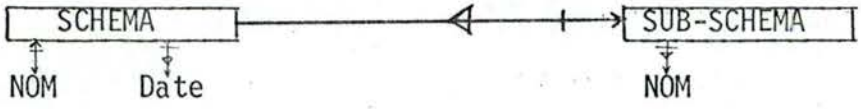
8.1. Génération des listings DDL

1. Génération de la clause "SUB-SCHEMA"

Soit la clause

SUB-SCHEMA NAME is <sub-schema-name> OF SCHEMA <schema-name>.

Graphe d'accès



Le générateur DDL accède à l'objet SCHEMA par "nom"."Date" représente la date de génération.

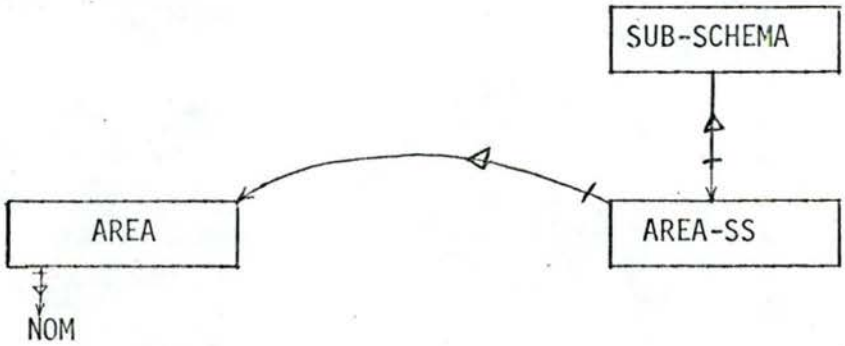
Le nom d'un SCHEMA est un identifiant pour avoir un accès rapide, nous proposons d'accéder à SCHEMA par son nom. Nous ajoutons l'item Date pour tenir compte des modifications possibles d'un SCHEMA DBMS-20.

2. Génération de la clause "AREA"

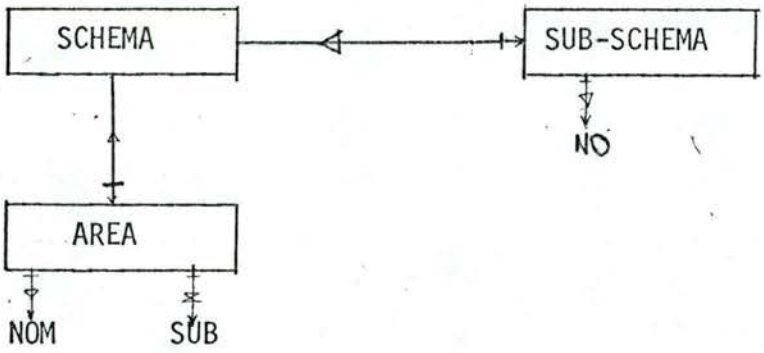
Soit la Clause

AREA NAME IS <area-name>.

Graphe d'accès



Pour simplifier, nous proposons un graphe d'accès comme suit :

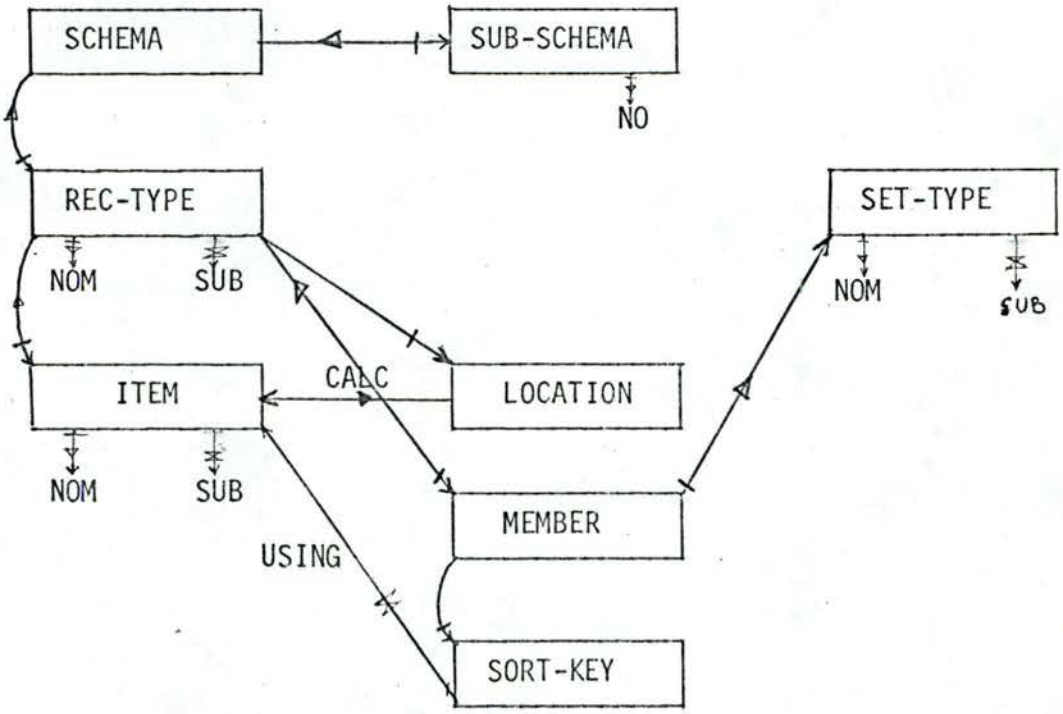


Chaque SUB-SCHEMA d'un SCHEMA est identifié par un numéro interne, relative à un SCHEMA. Chaque AREA a une liste ("sub") des numéros SUB-SCHEMA qui utilisent cette AREA. Pour générer cette clause, il suffit de parcourir la liste "sub" et de vérifier s'il existe dans cette liste le numéro ("NO") du SUB-SCHEMA concerné.

3. Génération de la clause "RECORD"

Soit la clause
RECORD NAME IS <record-name> [RETRIEVAL ONLY]

Graphe d'accès



La clause RETRIEVAL ONLY est générée lorsque

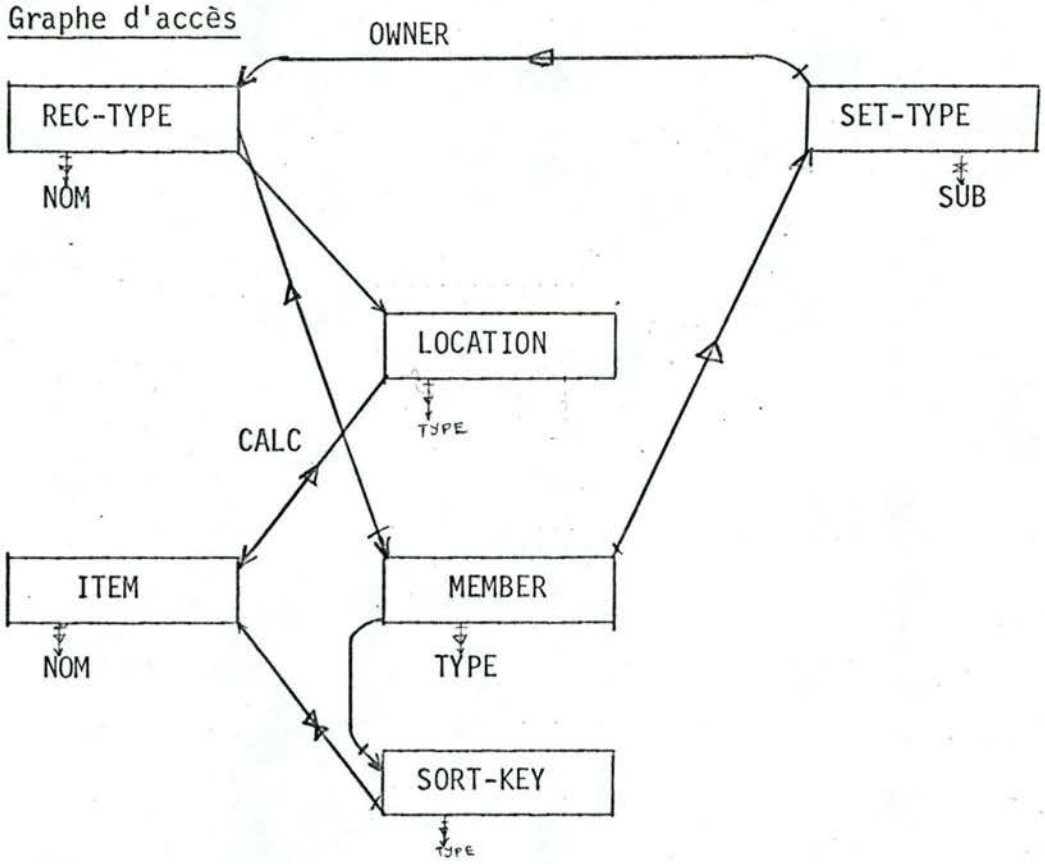
- un item est déclaré dans la LOCATION MODE CALC et n'appartient pas à ce type de record du sub-schéma concerné
- ce type de record est member d'un type de set du sub-schéma donné et un des items de la SORT-KEY ne lui appartient pas.

4. Génération de la clause IDENTIFIER

Soit la clause

IDENTIFIER IS <item-name-1> [<item-name-2>] $^{i}_{o}$

Graphe d'accès



S'il n'existe pas de RETRIEVAL ONLY

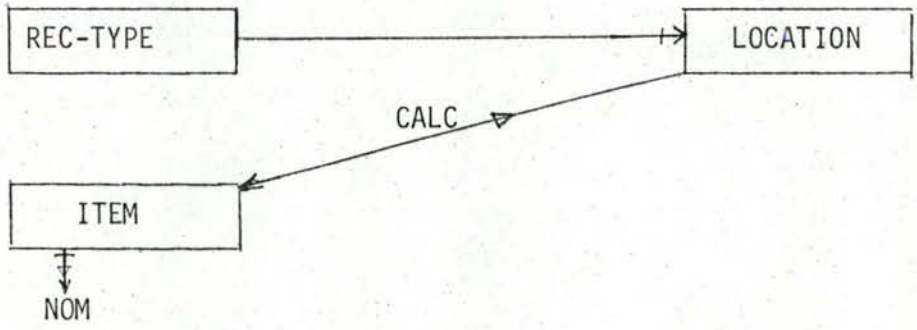
1. Oncherche LOCATION MODE CALC avec DUPLICATES NOT ALLOWED
   IDENTIFIER, dans ce cas, se compose de l'item AREA et des items de la relation CALC
2. Après si ce type de record est member MANDATORY AUTOMATIC d'un type de set (du sub-schéma spécifié) dont le type owner est SYSTEM et il existe un SORT-KEY DUPLICATES NOT ALLOWED. IDENTIFIER dans ce cas, se compose des items participant à la SORT-KEY.

5. Génération de la clause ACCESS-KEY

Soit la clause

ACCESS-KEY is <item-name-1> [<item-name-2>]$_{1}^{i}$

Graphe d'accès



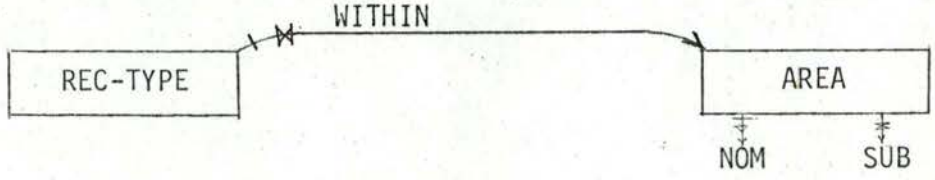
S'il n'existe pas de RETRIEVAL ONLY, on cherche LOCATION MODE CALC. ACCESS-KEY se compose de l'item AREA et des items de la relation CALC.

6. Génération de la clause "AREA"

Soit la clause

02 AREA PIC X(30) VALUE <area-name-1> [<area-name-2>]$_{o}^{i}$

Graphe d'accès



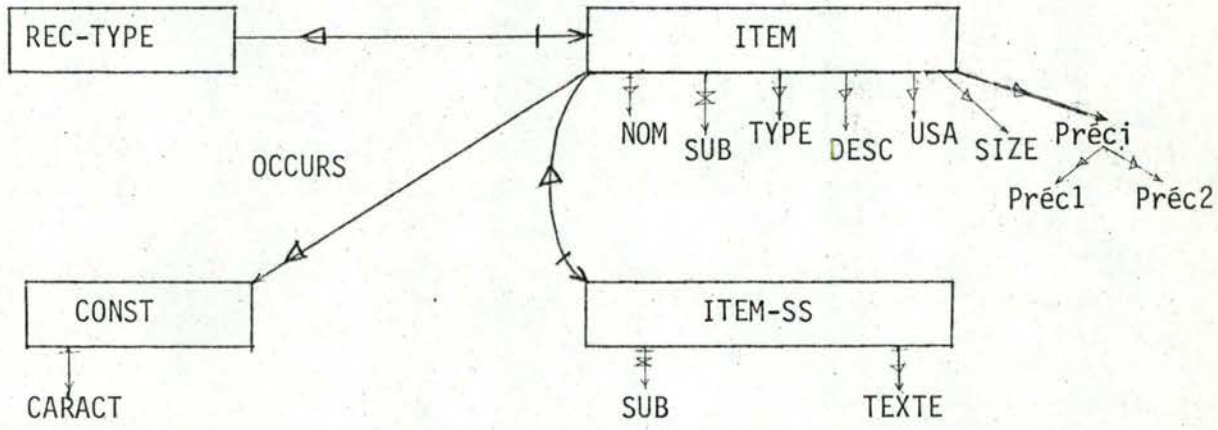
Par la relation WITHIN, on trouve les AREAS associés à un type de record, par la liste "sub", on sélectionne les AREAS concernés.

7. Génération de la description d'item

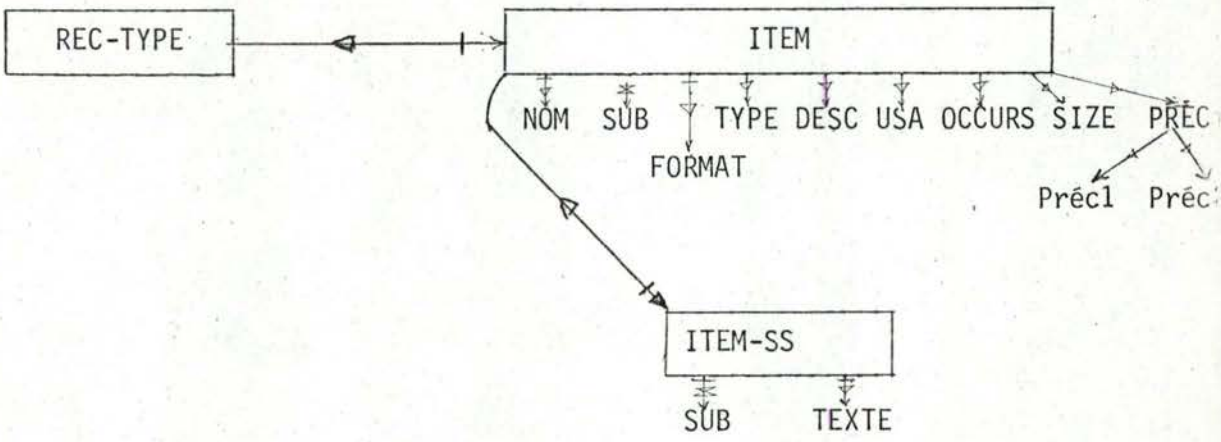
Soit la description d'un item :

<level-number> <data-name>
[PICTURE IS <picture-string>]
[USAGE IS { COMP
COMP-1
COMP-3
DISPLAY-6
DISPLAY-7
DISPLAY-9
DB-KEY }]
[OCCURS <integer> TIMES]

Graphe d'accès



ITEM a toujours le niveau 02, ce qui permet de le simplifier.
La relation OCCURS est une caractéristique d'items, pour simplifier nous considérons cette relation comme un item de l'objet ITEM.
Nous ajoutons un item FORMAT pour mieux distinguer les 3 formats du DDL-DBMS-20.



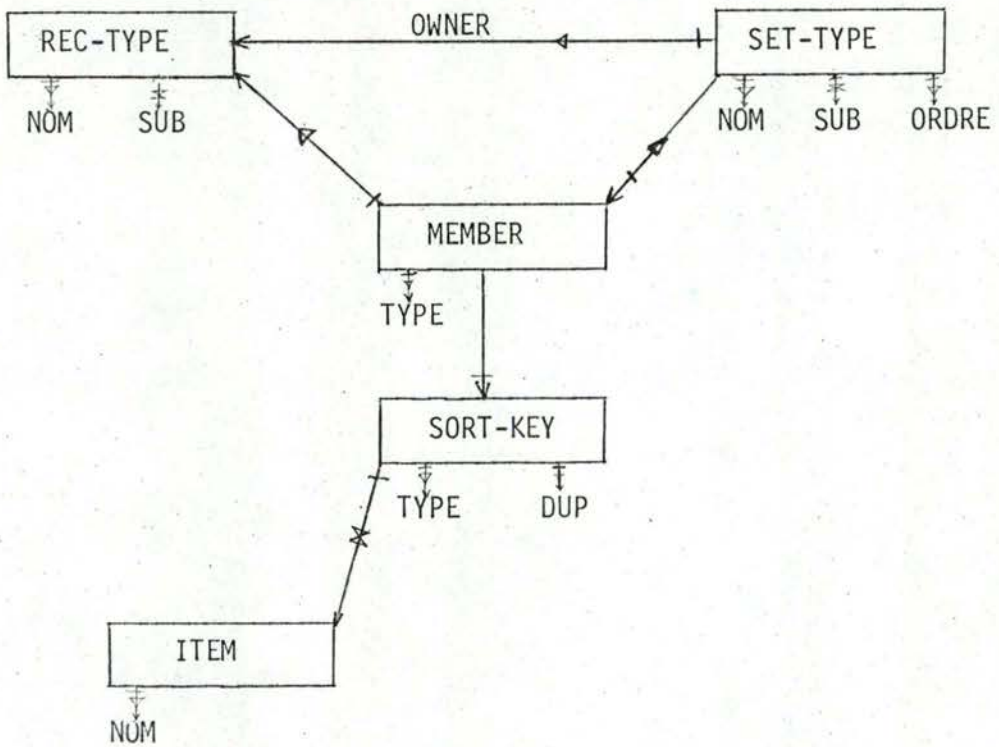
8. Génération de la description de type de set

Soit la description d'un type de set

SET NAME IS <set-name>
OWNER IS { <record-name-1> / SYSTEM }
MEMBER IS <record-name-2> { MANDATORY / OPTIONAL } { AUTOMATIC / MANUAL }

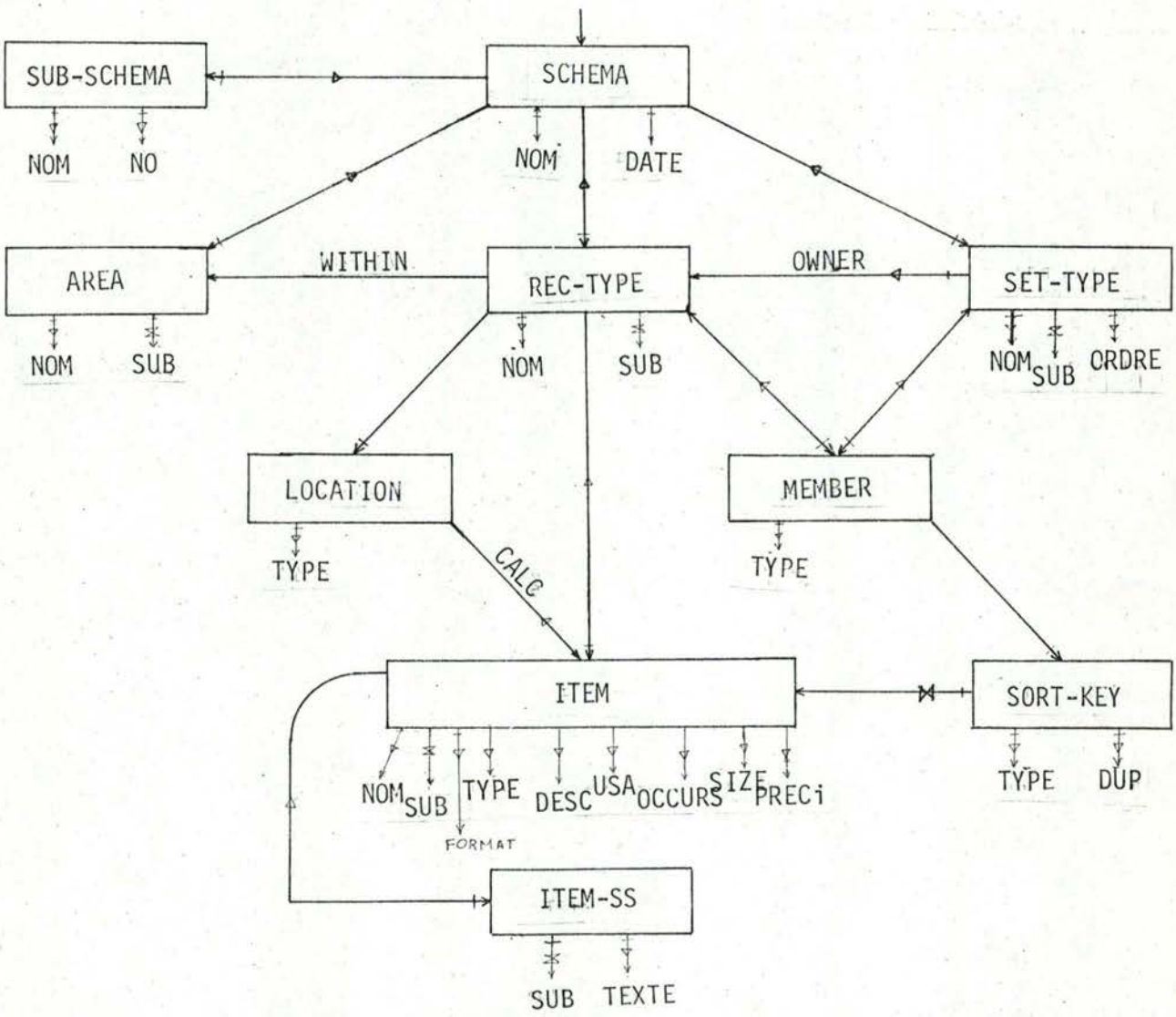
```
[ IDENTIFIER OF SET IS  <item-name-1>[ <item-name-2> ]i
                                                     o
[ ORDER IS {FIRST}
           {LAST }
 ORDER IS SORTED BY DB-KEY
                    {ASCENDING } KEY IS <item-name-3> [ <item-name-4> ]j
                    {DESCENDING }                                     o
                                [DUPLICATES ARE {FIRST} ALLOWED]
                                               {LAST }
```

Graphe d'accès



9. Graphe d'accès de la génération des listings DDL

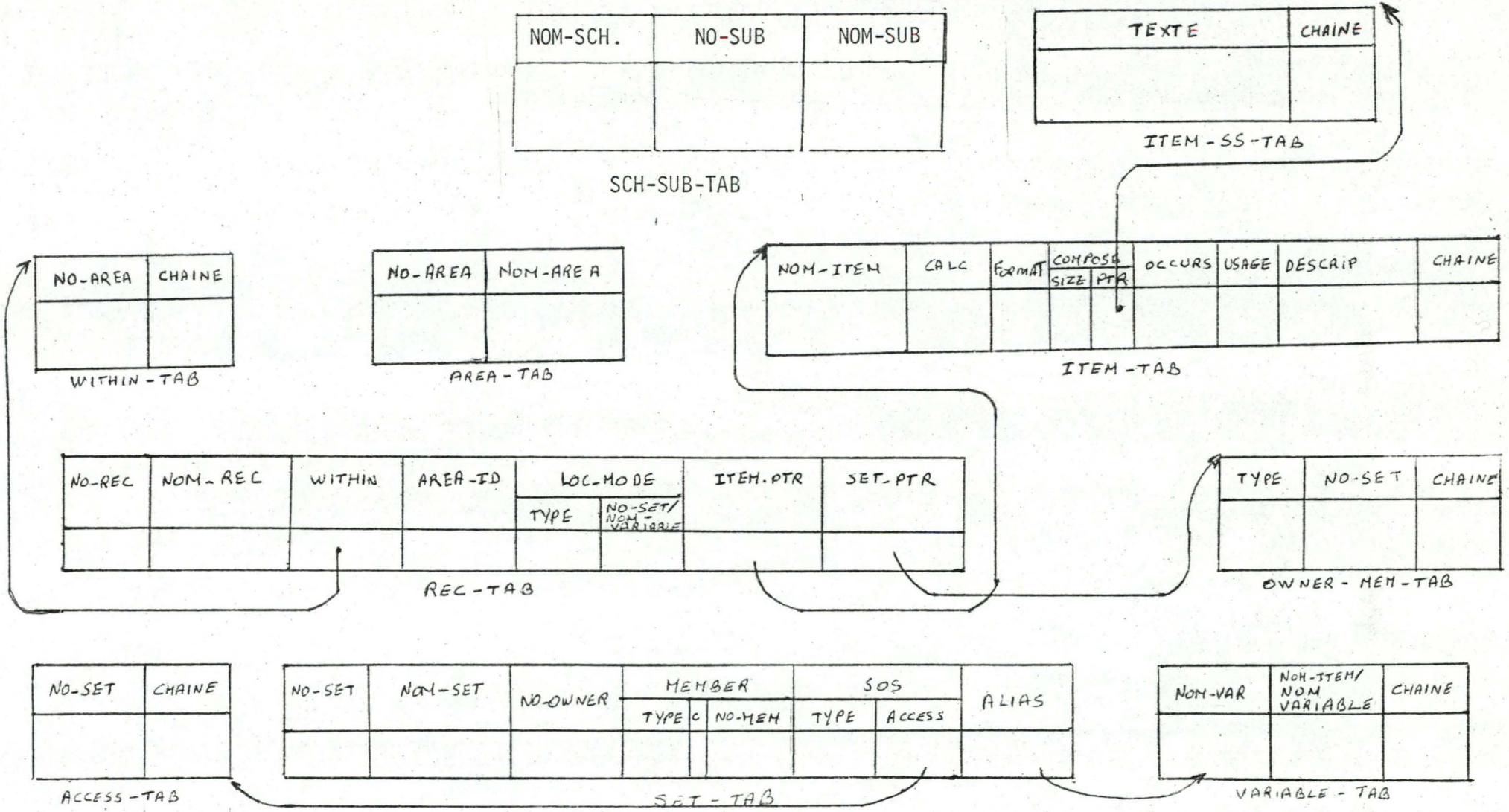
(Voir page suivante)



8.2. *Générations des tables pour DMLP*

La compilation d'un programme DML a besoin des tables internes pour établir des liaisons entre ce programme et le SGBD via un sub-schéma (d'un schéma) indiqué dans ce programme.

L'analyse des besoins du compilateur DML nous permet d'établir les tables suivantes :


NOM-SCH.
NO-SUB
NOM-SUB
SCH-SUB-TAB
TEXTE
CHAINE
ITEM-SS-TAB
NO-AREA
CHAINE
WITHIN-TAB
NO-AREA
NOM-AREA
AREA-TAB
NOM-ITEM
CALC
FORMAT
COMPOSE
SIZE
PTR
OCCURS
USAGE
DESCRIP
CHAINE
ITEM-TAB
NO-REC
NOM-REC
WITHIN
AREA-ID
LOC-MODE
TYPE
NO-SET/ NOM-VARIABLE
ITEM.PTR
SET_PTR
REC-TAB
TYPE
NO-SET
CHAINE
OWNER-MEM-TAB
NO-SET
CHAINE
ACCESS-TAB
NO-SET
NOM-SET
NO-OWNER
MEMBER
TYPE
C
NO-MEM
SOS
TYPE
ACCESS
ALIAS
SET-TAB
NOM-VAR
NOM-ITEM/ NOM VARIABLE
CHAINE
VARIABLE-TAB

-SCH-SUB-TAB contient tous les schémas et leurs sub-schéma que les utilisateurs peuvent employer.

A partir du sub-schéma d'un schéma indiqué dans le programme DML-MCS, le compilateur vérifie l'existence de ce sub-schéma en utilisant la SCH-SUB-TAB. S'il existe, le compilateur va appeler le fichier dont le nom est constitué de <nom-Sch><NO-Sub> (le nom du schéma et le numéro du subschéma). Ce fichier contient les tables suivantes :

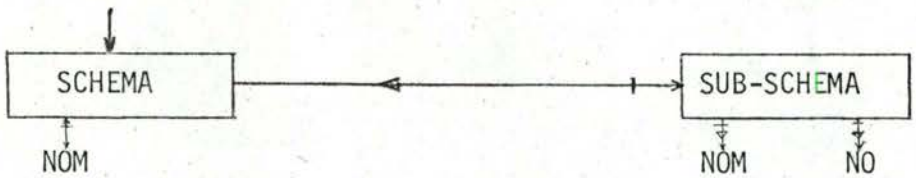
- AREA-TAB : contenant toutes les aréas du sub-schéma. Une arée est associée à un numéro interne (NO-AREA, ce numéro est créé lors du chargement de la BD)
- REC-TAB contenant tous les types de record du sub-schéma avec
  - un nom de type de record (NOM-REC) associé à son numéro interne (NO-REC)
  - un pointeur WITHIN qui pointe vers le début d'une chaine de numéros d'aréa. (WITHIN-TAB) (pour indiquer les aréas associées à un type de record).
  - AREA-ID est le nom de AREA-ID s'il existe
  - LOC-MODE est le LOCATION MODE
  - un pointeur ITEM-PTR qui pointe vers le début d'une chaîne d'items contenue dans ITEM-TAB (pour indiquer les items associés).
  - un pointeur SET-PTR qui pointe vers le début d'une chaîne de numéros de type de set contenue dans OWNER-MEM-TAB (pour indiquer les types de set auxquels ce type de record participe soit comme owner soit comme member).
- ITEM-TAB contenant tous les items associés aux types de record du sub-schéma

  Les items associés à un typede record forment une chaine :
- CALC pour indiquer les items du LOCATION MODE CALC (éventuellement)
- COMPOSE pour indiquer
  - la longueur d'un item composé (SIZE)
  - le début d'une chaine des descendants de l'item composé (PTR) contenus dans ITEM-SS-TAB
- DESCRIP pour décrire l'item (la clause PICTURE du COBOL)
- SET-TAB contenant tous les types de set du sub-schéma avec
  - un nom de type de set (NOM-SET) associé à son numéro interne (NO-SET)
  - le numéro du type de record qui est owner (NO-OWNER)
  - une chaîne des types member; chaque type member a :
    - le type (TYPE) indiquant les propriétés MANDATORY/OPTIONAL et AUTOMATIC/MANUAL
    - le numéro du type de record qui est member (NO-MEM)
  - un mode de sélection de set (SOS) avec :
    . le type de SOS
    . le pointeur ACCESS qui pointe vers le début d'une chaîne de chemin d'accès (ACCESS-TAB).

. le pointeur ALIAS qui pointe vers le début d'une chaîne de variable (VARIABLE-TAB) cette variable est associée à un NOM-ITEM ou NOM-VAR (nom de variable) pour décrire la clause ALIAS

8.2.1. Les graphes d'accès

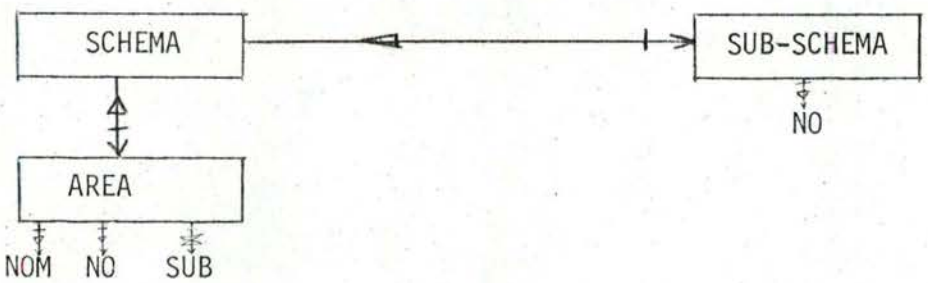
1. Générations de SCH-SUB-TAB



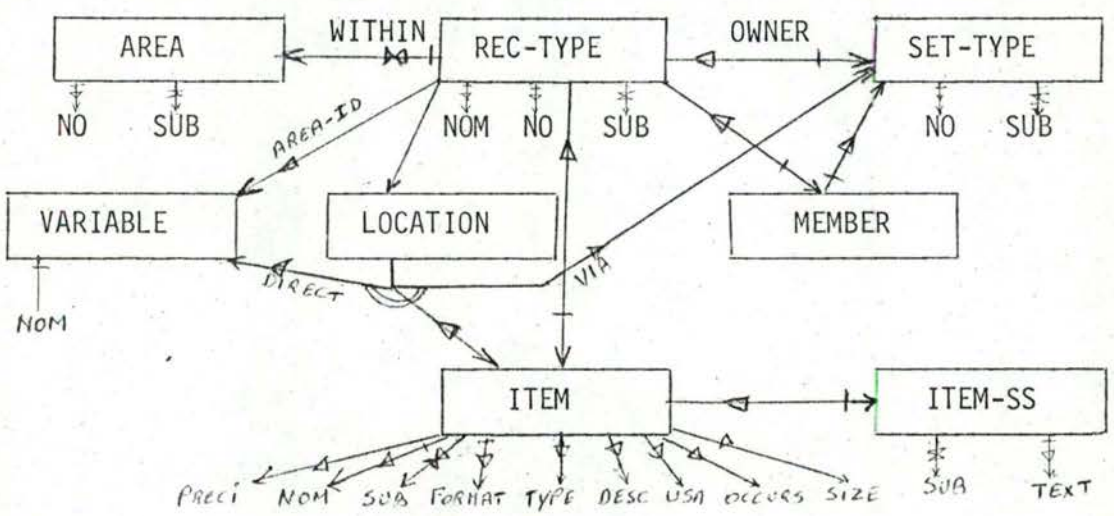
Il y a 2 cas

- on veut générer les tables de tous les SCHEMAS et SUB-SCHEMAS
- on veut générer les tables d'un SCHEMA ou un SUB-SCHEMA particulier, ce qui justifie l'accès de NOM vers SCHEMA

2. Génération de AREA-TAB



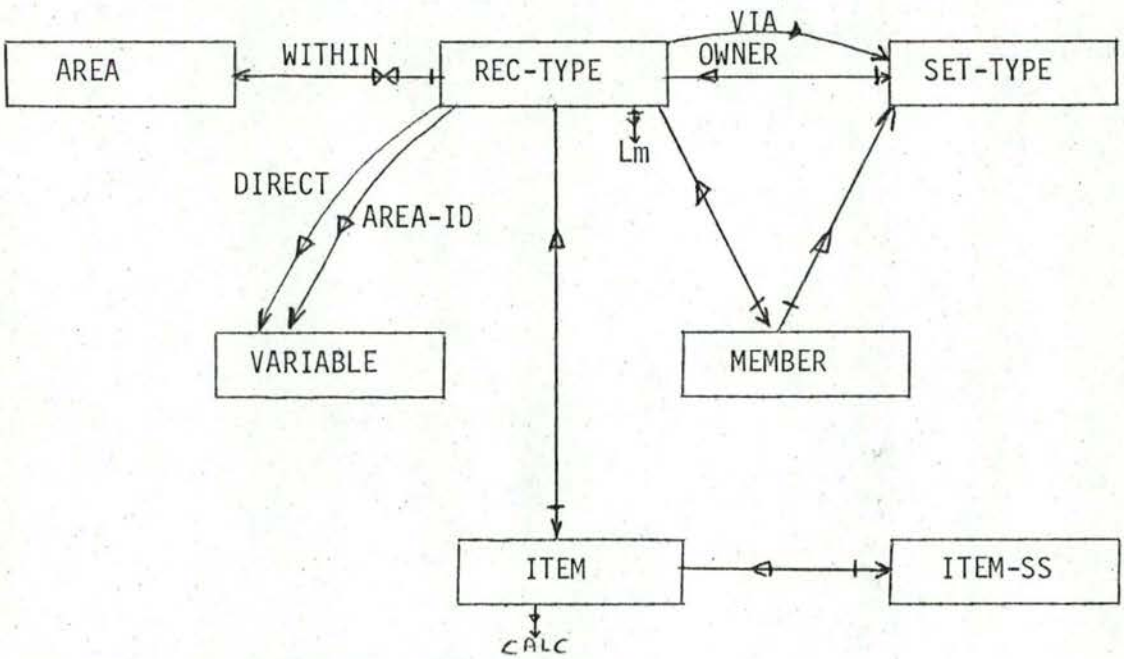
3. Génération de REC-TAB, ITEM-TAB, WITHIN-TAB, OWNER-MEM-TAB, VARIABLE-TAB



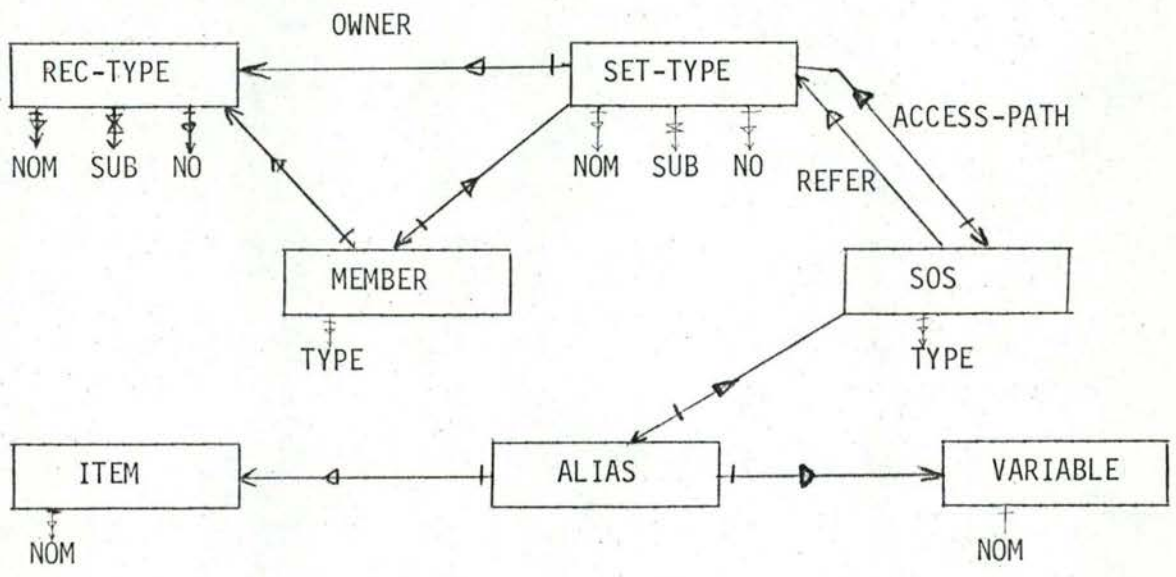
Pour ITEM-TAB, on doit transformer des types d'items en ceux du COBOL, ce qui entraîne un traitement des valeurs d'ITEM et d'ITEM-SUB.

Les items utilisés dans LOCATION MODE CALC doivent être ceux du type de record déclaré ainsi, ce qui nous permet de remplacer la relation CALC (LOCATION,ITEM en un item de l'objet ITEM (item "calc")
LOCATION MODE est une propriété propre à un type de record, ainsi, on peut simplifier la relation (REC-TYPE, LOCATION) en ajoutant un item de REC-TYPE (item "Lm"), qui indique les 3 modes de LOCATION.

Après les simplifications, on a le graphe suivant :

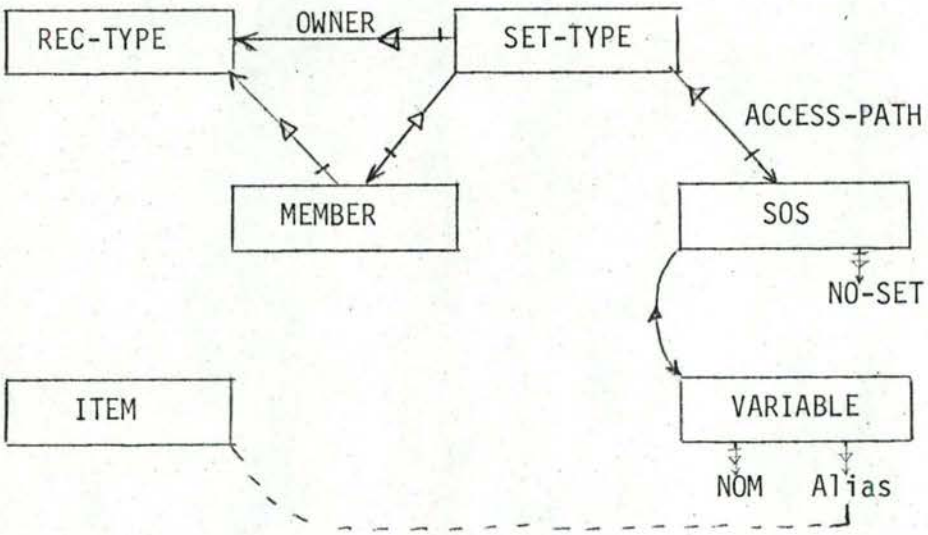


4. Génération de SET-TAB, ACCESS-TAB, VARIABLE-TAB



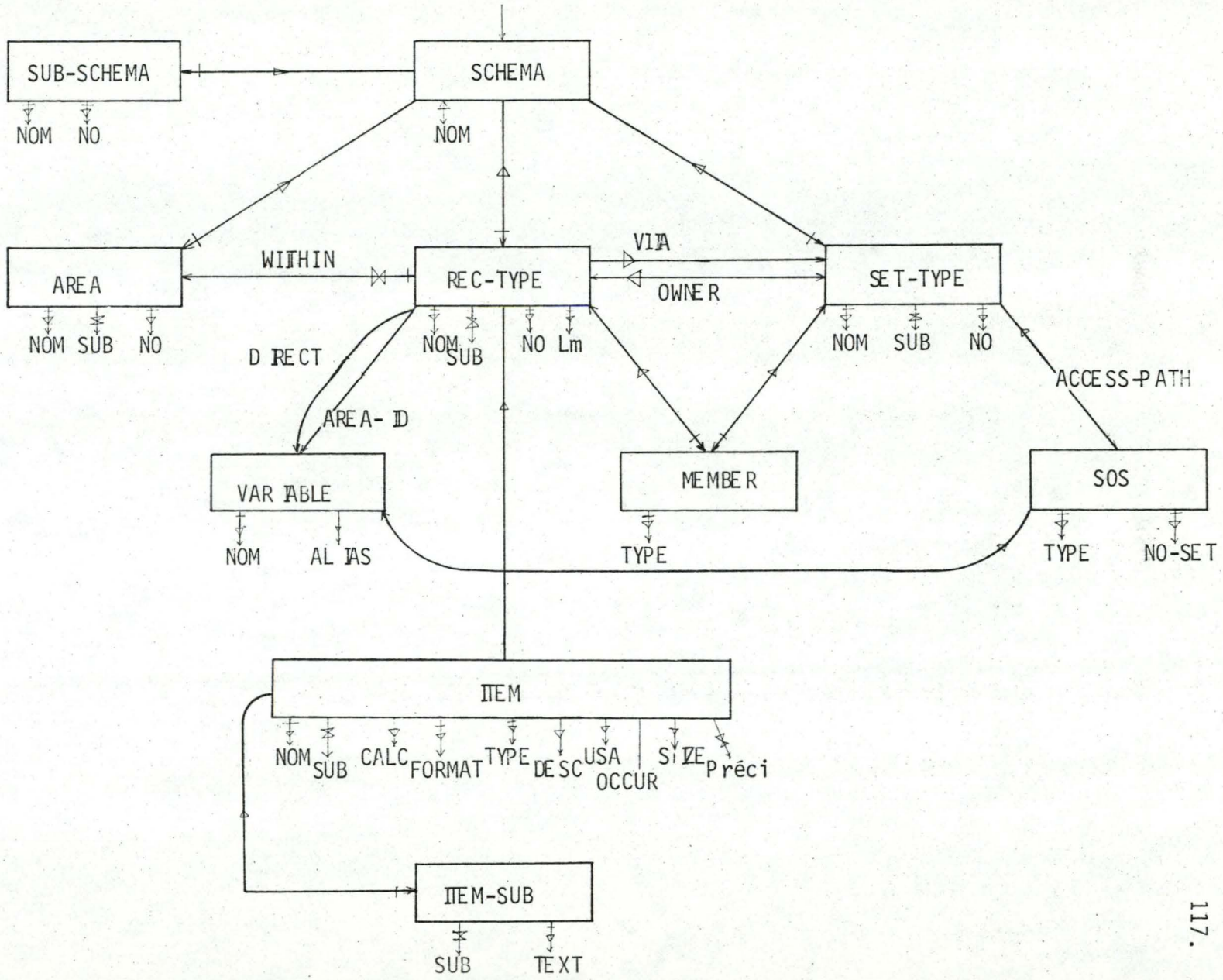
On peut simplifier la relation REFER en ajoutant un item "NO-SET" dans SOS. Nous travaillons sur des schémas DBMS-20 qui sont corrects syntaxiquement et sémantiquement, ce qui nous permet de simplifier objet ALIAS et ses relations en ajoutant un item "Alias" dans VARIABLE ("Alias" référencie à un ITEM). La relation ALIAS (SOS, ITEM, VARIABLE) n'accepte pas une telle simplification, mais ici, nous ne vérifions pas l'unicité du nom de VARIABLE (ce qui n'est pas nécessaire dans ce cas).

Nous avons donc le graphe suivant :



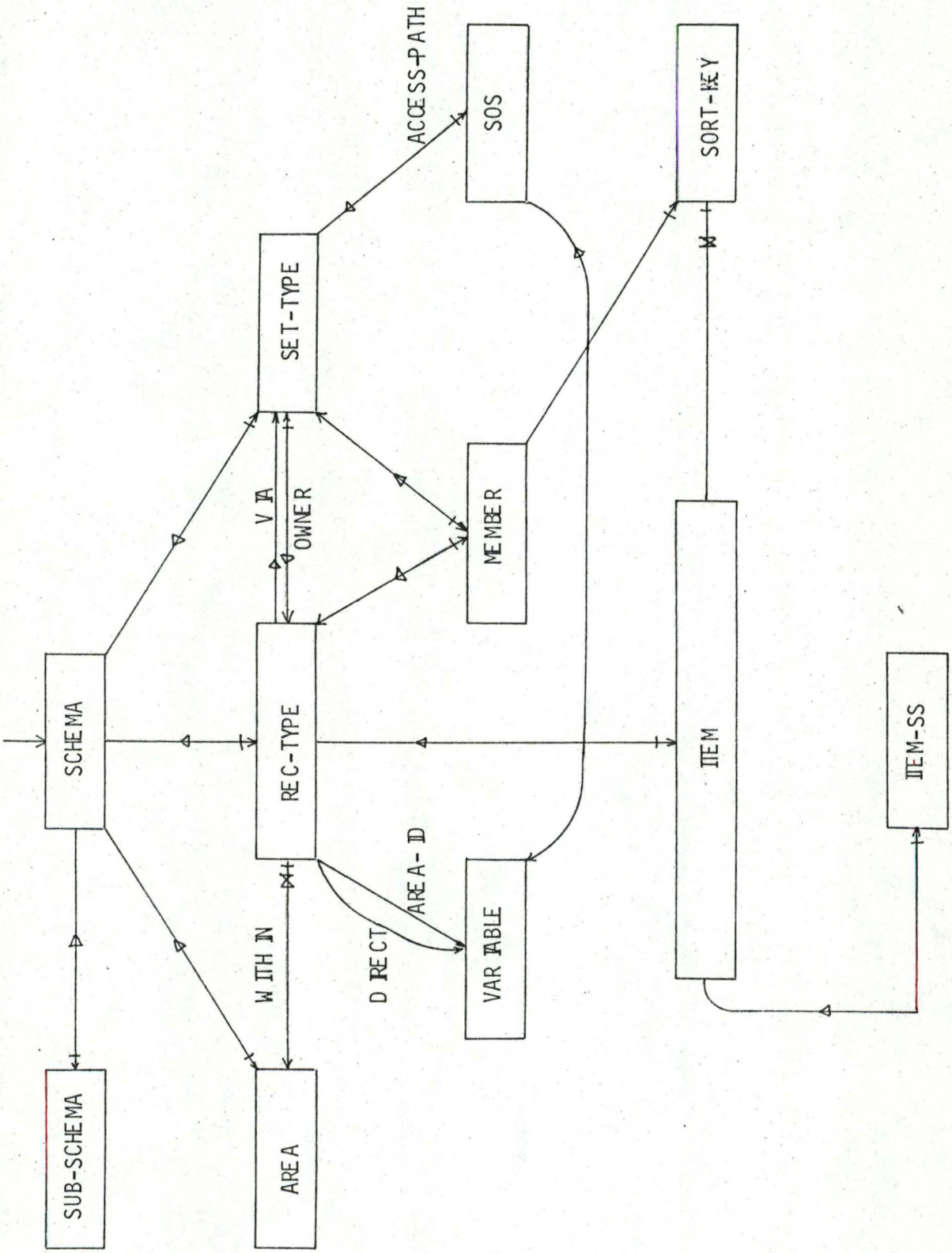
5. Graphe d'accès pour DML du modèle BMS-20 simplifié

Puisque nous ne vérifions pas l'unicité du nom de VARIABLE, les relations DIRECT(REC-TYPE, VARIABLE) et AREA-ID (REC-TYPE, VARIABLE) deviennent "one-to-one" et on peut avoir plusieurs "NOM" de VARIABLE de même valeur.


SUB-SCHEMA
NOM
NO
SCHEMA
NOM
AREA
NOM
SUB
NO
WITHIN
REC-TYPE
NOM
SUB
NO
Lm
VIA
OWNER
SET-TYPE
NOM
SUB
NO
DIRECT
AREA-ID
ACCESS-PATH
VARIABLE
NOM
ALIAS
MEMBER
TYPE
SOS
TYPE
NO-SET
ITEM
NOM
SUB
CALC
FORMAT
TYPE
DESC
USA
OCCUR
SIZE
Préci
ITEM-SUB
SUB
TEXT

8.3. Graphes d'accès complet

Les items sont présentés dans les 2 graphes DDL et DML.



CHAPITRE IX : IMPLEMENTATION DU SCHEMA

9.1. Système DBMS-20

Afin de déterminer les paramètres physiques en vue de l'implémentation du schéma, nous citons ici un nombre de caractéristiques du système DBMS-20.

1. DEC-20 offre 3 modes de représentation des valeurs des données : 6-BIT (DISPLAY-6), 7-BIT (DISPLAY-7 ou ASCII), 9-BIT (DISPLAY-9 ou EBCDIC)
2. Les records d'une BD DBMS-20 sont groupés en pages de longueur déterminée par l'administrateur (option par défaut : 512 mots de 36 bits). Chaque aréa est divisée en un nombre de pages.
3. Au niveau d'une aréa, on peut déterminer :
   - la longueur de page d'une aréa
   - le nombre des tampons (BUFFER)
   - le nombre de chaînes CALC pour chaque page
   - les pages dans lesquelles les records d'un certain type peuvent être enregistrés (RANGE)
   - le nombre de record par page
4. Au niveau d'un type de record, on détermine
   - LOCATION MODE
   - Les aréas dans lesquelles se trouvent des records
5. Au niveau d'un type de set, on détermine :
   - le nombre de pointeurs inverses (LINKED TO OWNER ...)
   - l'ordre de set
   - les caractéristiques d'un type member (AUTOMATIC/MANUAL ...)
   - la SOS
6. On peut déterminer des journaux pour "BACKUP/RECOVERY"

Les contraintes du DBMS-20

Dans DBMS-20, on ne peut définir au maximum que :

1. - 36 sub-schémas/schéma
2. 480 types de record/schéma
3. 512 types de set/type de record.

En effet, chaque record a 512 pointeurs au plus

Le nombre d'aréas n'est pas fixé explicitement. On considère une aréa comme un fichier.

Les autres restrictions sont indiquées au chapitre V.

9.2. *Implémentation du schéma*

Nous transformons la relation m-n WITHIN (AREA, REC-TYPE) en un typ de record WITH-REC avec l'item NO (numéro d'aréa) et une relation REC-WITH, en effet, on n'a pas besoin de relation inverse (de AREA vers REC-TYPE) et lors du chargement de la BD, la recherche de numéros d'aréas (pour enregistrer dans WITH-REC) ne prend pas beaucoup de temps (le nombre d'aréa d'un schéma n'est pas grand en général).

D'autre part, pour diminuer le nombre de type de sets, nous implémentons les trois relations DIRECT, VIA, AREA-ID par 3 items du REC-TYPE
- LM qui indique le LOCATION MODE (DIRECT, CALC ou VIA)
- IDLM qui est une référence à un type de set (LOCATION MODE VIA) ou à une variable (LOCATION MODE DIRECT)
- IDAREA qui est une référence à AREA-ID

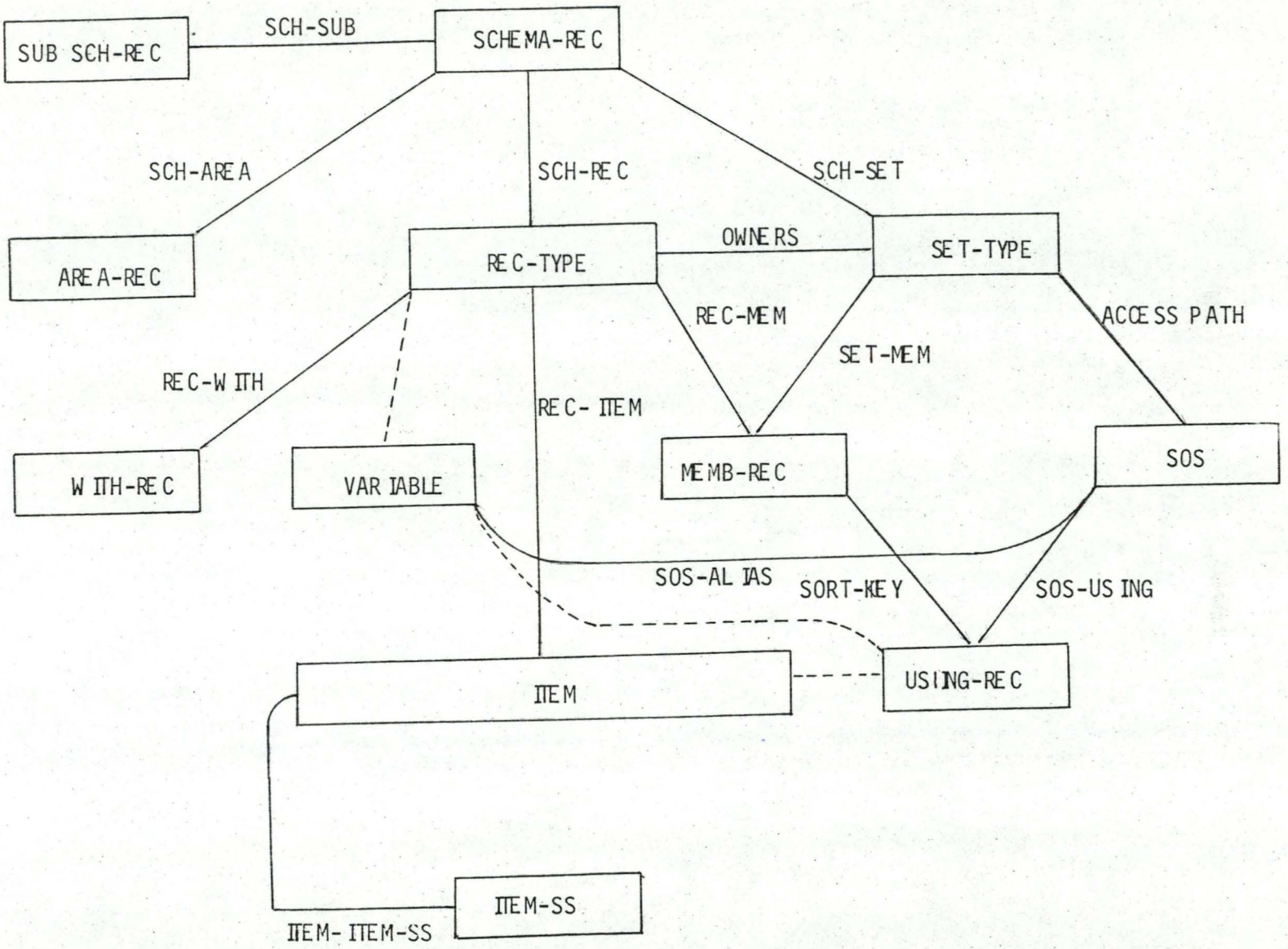
Nous ajoutons aussi dans SET-TYPE un item SYS pour indiquer si un type owner est SYSTEM ce qui rend "faible" la relation OWNERS(SET-TYPE, REC-TYPE

En tenant compte des mots clés du DBMS-20, nous devons changer des noms d'objets ou de relations.

En vue d'une extension possible des autres applications, nous faisons une implémentation complète de toute description de schéma DBMS-20.

1. Graphe d'implémentation

(Les items sont présentés dans la description du schéma).


SUB SCH-REC
SCH-SUB
SCHEMA-REC
SCH-AREA
SCH-REC
SCH-SET
AREA-REC
REC-TYPE
OWNERS
SET-TYPE
REC-MEM
ACCESS PATH
SET-MEM
REC-WITH
REC- ITEM
WITH-REC
VARIABLE
MEMB-REC
SOS
SOS-ALIAS
SORT-KEY
SOS-USING
ITEM
USING-REC
ITEM-SS
ITEM- ITEM-SS

2. Description du schéma

```
00100    IMAGES IN ORDER BY COMMAND.
00200    MODE ALL.
00300    INTERCEPT ALL.
00400    RPP IS  170.
00500    ASSIGN DDLDML-AREA TO FICJ01
00600    BUFFER 10
00700    FIRST PAGE IS 1
00800    LAST PAGE IS 200
00900    PAGE SIZE IS 512 WORDS.
01000
01100
01200    SCHEMA NAME IS DBMS0.
01300
01400    AREA NAME IS DDLDML-AREA PRIVACY LOCK FOR UPDATE SPHX0.
01500
01600    RECORD NAME IS SCHEMA-REC
01700             LOCATION MODE IS CALC USING SCH-NOM DUPLICATES NOT
01800             WITHIN DDLDML-AREA.
01900    02  SCH-NOM       PIC X(6) USAGE DISPLAY-7.
02000    02  SCH-DATE      PIC X(6) USAGE DISPLAY-7.
02100
02200    RECORD NAME IS SUBSCH-REC
02300              LOCATION MODE IS VIA SCH-SUB
02400              WITHIN DDLDML-AREA.
02500    02  SUB-NOM              PIC X(30) USAGE DISPLAY-7.
02600    02  SUB-CLE              PIC X(5) USAGE DISPLAY-7.
02700    02  SUB-NO               PIC 9(2) USAGE DISPLAY-7.
02800
02900    RECORD NAME IS AREA-REC
03000              LOCATION MODE IS VIA SCH-AREA
03100              WITHIN DDLDML-AREA.
03200    02  ARE-SUB          PIC X  USAGE DISPLAY-7 OCCURS 36.
03300    02  ARE-NOM          PIC X(30) USAGE DISPLAY-7.
03400    02  ARE-CLE          PIC X(5) USAGE DISPLAY-7 OCCURS 6.
03500    02  ARE-NO           PIC 9(3) USAGE DISPLAY-7.
03600    02  ARE-TEMP         PIC 9 USAGE DISPLAY-7 OCCURS 36.
03700
03800    RECORD NAME IS REC-TYPE
03900              LOCATION MODE IS VIA SCH-REC
04000              WITHIN DDLDML-AREA.
04100    02  REC-SUB              PIC X USAGE DISPLAY-7 OCCURS 36.
04200    02  REC-NOM              PIC X(30) USAGE DISPLAY-7.
04300    02  REC-LM               PIC 9 USAGE DISPLAY-7.
04400    02  REC-IDLM             TYPE IS DBKEY.
04500    02  REC-IDAREA           TYPE IS DBKEY.
04600
04700    RECORD NAME IS SET-TYPE
04800              LOCATION MODE IS VIA SCH-SET
04900              WITHIN DDLDML-AREA.
05000    02  SET-SUB          PIC X USAGE DISPLAY-7 OCCURS 36.
05100    02  SET-NOM          PIC X(30) USAGE DISPLAY-7.
05200    02  SET-ORDRE        PIC 99 USAGE DISPLAY-7.
05300    02  SET-SYS          PIC 9 USAGE DISPLAY-7.
05400    02  SET-NO           PIC 9(3) USAGE DISPLAY-7.
05500    02  SET-LINK         PIC 9 USAGE DISPLAY-7.
05600
05700    RECORD NAME IS  WITH-REC
05800              LOCATION MODE IS VIA REC-WITH
05900              WITHIN DDLDML-AREA.
06000    02  WIT-NO           PIC 9(3) USAGE DISPLAY-7.
```

```
05100
06200    RECORD NAME IS MEMB-REC
06300              LOCATION MODE IS VIA SET-MEM
06400              WITHIN DDLDML-AREA.
06500    02  MEM-TYPE          PIC 9 USAGE DISPLAY-7.
06600    02  MEM-SORT          PIC 9 USAGE DISPLAY-7.
06700    02  MEM-SORT-DUP      PIC 9 USAGE DISPLAY-7.
06800    02  MEM-LINK          PIC 9 USAGE DISPLAY-7.
06900
07000    RECORD NAME IS SDS
07100              LOCATION MODE VIA ACCESS-PATH
07200              WITHIN DDLDML-AREA.
07300    02  SDS-NO            PIC 9(3) USAGE DISPLAY-7.
07400
07500    RECORD NAME IS ITEM
07600              LOCATION MODE VIA REC-ITEM
07700              WITHIN DDLDML-AREA.
07800    02  ITE-SUB           PIC X USAGE DISPLAY-7 OCCURS 36.
07900    02  ITE-NOM           PIC X(30) USAGE DISPLAY-7.
08000    02  ITE-FORMAT        PIC 9 USAGE DISPLAY-7.
08100    02  ITE-TYPE          PIC 9 USAGE DISPLAY-7 OCCURS 3.
08200    02  ITE-CALC          PIC 9 USAGE DISPLAY-7.
08300    02  ITE-USA           PIC 9 USAGE DISPLAY-7.
08400    02  ITE-SIZE          PIC 9999 USAGE DISPLAY-7.
08500    02  ITE-OCCUR         PIC 9(6) USAGE DISPLAY-7.
08600    02  ITE-PREC1         PIC 9(3) USAGE DISPLAY-7.
08700    02  ITE-PREC2         PIC 9(3) USAGE DISPLAY-7.
08800    02  ITE-DESC          PIC X(30) USAGE DISPLAY-7.
08900
09000    RECORD NAME IS USING-REC
09100              LOCATION MODE VIA SORT-KEY
09200              WITHIN DDLDML-AREA.
09300    02  USI-TYPE          PIC 9 USAGE DISPLAY-7.
09400    02  USI-ID            TYPE DBKEY.
09500
09600    RECORD NAME IS VARIABLE
09700              LOCATION MODE DIRECT ID-VAR
09800              WITHIN DDLDML-AREA.
09900    02  VAR-NOM           PIC X(30) USAGE DISPLAY-7.
10000    02  VAR-ALIAS         TYPE IS DBKEY.
10100
10200    RECORD NAME IS ITEM-SS
10300              LOCATION MODE VIA ITEM-ITEM-SSUB
10400              WITHIN DDLDML-AREA.
10500    02  ISS-SUB           PIC X USAGE DISPLAY-7 OCCURS 36.
10600    02  ISS-TEXTE         PIC X(120) USAGE DISPLAY-7.
10700
10800    SET NAME IS SCH-SUB
10900        MODE IS CHAIN
11000        ORDER ALWAYS FIRST
11100        OWNER IS SCHEMA-REC.
11200        MEMBER IS SUBSCH-REC MANDATORY AUTOMATIC
11300        SELECTION CURRENT.
11400
11500    SET NAME IS SCH-AREA
11600        MODE IS CHAIN
11700        ORDER IS ALWAYS FIRST
11800        OWNER IS SCHEMA-REC
11900        MEMBER IS AREA-REC MANDATORY AUTOMATIC
12000        SELECTION IS CURRENT.
```

```
12100
12200    SET NAME IS  SCH-REC
12300        MODE IS CHAIN
12400        ORDER IS ALWAYS FIRST
12500        OWNER IS  SCHEMA-REC
12600        MEMBER IS REC-TYPE MANDATORY AUTOMATIC
12700        SELECTION IS CURRENT.
12800
12900    SET NAME IS SCH-SET
13000        MODE IS CHAIN
13100        ORDER IS ALWAYS FIRST
13200        OWNER IS  SCHEMA-REC
13300        MEMBER IS SET-TYPE MANDATORY AUTOMATIC
13400        SELECTION IS CURRENT.
13500
13600    SET NAME IS REC-WITH
13700        MODE IS CHAIN
13800        ORDER IS ALWAYS FIRST
13900        OWNER IS  REC-TYPE
14000        MEMBER IS  WITH-REC MANDATORY AUTOMATIC
14100        SELECTION CURRENT.
14200
14300    SET NAME IS REC-ITEM
14400        MODE IS CHAIN
14500        ORDER IS ALWAYS FIRST
14600        OWNER IS  REC-TYPE
14700        MEMBER IS ITEM MANDATORY AUTOMATIC
14800        SELECTION IS CURRENT.
14900
15000    SET NAME IS ITEM-ITEM-SS/UB
15100        MODE IS CHAIN
15200        ORDER IS ALWAYS FIRST
15300        OWNER IS ITEM
15400        MEMBER IS ITEM-SS MANDATORY AUTOMATIC
15500        SELECTION IS  CURRENT.
15600
15700    SET NAME IS  OWNERS
15800        MODE IS CHAIN
15900        ORDER IS ALWAYS FIRST
16000        OWNER IS  REC-TYPE
16100        MEMBER IS  SET-TYPE OPTIONAL MANUAL
15200        LINKED TO OWNER
16300        SELECTION CURRENT.
16400
16500    SET NAME IS REC-MEM
16600        MODE IS CHAIN
16700        ORDER IS ALWAYS FIRST
16800        OWNER IS  REC-TYPE
16900        MEMBER IS MEMB-REC MANDATORY AUTOMATIC
17000        LINKED TO OWNER
17100        SELECTION IS CURRENT.
17200
17300    SET NAME IS SET-MEM
17400        MODE IS CHAIN
17500        ORDER IS ALWAYS FIRST
17600        OWNER IS SET-TYPE
17700        MEMBER IS  MEMB-REC MANDATORY AUTOMATIC
17800        LINKED TO OWNER
17900        SELECTION IS CURRENT.
18000
```

```
18100    SET NAME IS ACCESS-PATH
18200        MODE IS CHAIN
18300        ORDER IS ALWAYS FIRST
18400        OWNER IS SET-TYPE
18500        MEMBER IS SOS MANDATORY MANUAL
18600        SELECTION IS CURRENT.
18700
18800    SET NAME IS SOS-ALIAS
18900        MODE IS CHAIN
19000        ORDER IS ALWAYS NEXT
19100        OWNER IS SOS
19200        MEMBER IS VARIABLE OPTIONAL MANUAL
19300        SELECTION IS CURRENT.
19400
19500    SET NAME IS  SORT-KEY
19600        MODE IS CHAIN
19700        ORDER IS ALWAYS NEXT
19800        OWNER IS  MEMB-REC
19900        MEMBER IS  USING-REC  MANDATORY MANUAL
20000        SELECTION IS CURRENT.
20100
20200    SET NAME IS SOS-USING
20300        MODE IS CHAIN
20400        ORDER IS ALWAYS NEXT
20500        OWNER IS SOS
20600        MEMBER IS USING-REC MANDATORY MANUAL
20700        SELECTION CURRENT.
20800
20900
21000    SUB-SCHEMA NAME IS DBMS1 PRIVACY LOCK IS SPHX1.
21100    AREA SECTION.
21200        COPY ALL AREAS.
21300    RECORD SECTION.
21400        COPY ALL RECORDS.
21500    SET SECTION.
21600        COPY ALL SETS.
21700
21800    SUB-SCHEMA NAME IS DDLP PRIVACY LOCK IS SPHX2.
21900    AREA SECTION.
22000        COPY DDLDML-AREA.
22100    RECORD SECTION.
22200    01 SCHEMA-REC.
22300    01 SUBSCH-REC.
22400       02 SUB-NAM.
22500       02 SUB-NO.
22600    01 AREA-REC.
22700       02 ARE-SUB.
22800       02 ARE-NAM.
22900       02 ARE-NO.
23000    01 REC-TYPE.
23100       02 REC-SUB.
23200       02 REC-NAM.
23300       02 REC-LM.
23400    01 SET-TYPE.
23500       02 SET-SUB.
23600       02 SET-NAM.
23700       02 SET-ORDRE.
23800       02 SET-SYS.
23900          88 SYS VALUE 1.
24000    01 WITH-REC.
```

```
24100    01 MEMB-REC.
24200       02 MEM-TYPE.
24300       02 MEM-SORT.
24400       02 MEM-SORT-DUP.
24500    01 ITEM.
24600    01 USING-REC.
24700    01 VARIABLE.
24800       02 VAR-NUM.
24900    01 ITEM-SS.
25000    SET SECTION.
25100        COPY SCH-SUB SCH-AREA SCH-REC SCH-SET
25200             REC-WITH REC-ITEM REC-MEM
25300             SORT-KEY
25400             ITEM-ITEM-SS
25500             OWNERS SET-MEM.
25600
25700    SUB-SCHEMA NAME IS DMLP PRIVACY LOCK IS SPHX3.
25800    AREA SECTION.
25900         COPY DDLDML-AREA.
26000    RECORD SECTION.
26100    01 SCHEMA-REC.
26200       02 SCH-NUM.
26300    01 SUBSCH-REC.
26400       02 SUB-NUM.
26500       02 SUB-ND.
26600    01 AREA-REC.
26700       02 ARE-SUB.
26800       02 ARE-NUM.
26900       02 ARE-ND.
27000    01 REC-TYPE.
27100    01 SET-TYPE.
27200       02 SET-SUB.
27300       02 SET-NUM.
27400       02 SET-SYS.
27500       02 SET-ND.
27600    01 WITH-REC.
27700    01 MEMB-REC.
27800       02 MEM-TYPE.
27900    01 SDS.
28000    01 ITEM.
28100    01 USING-REC.
28200    01 VARIABLE.
28300    01 ITEM-SS.
28400    SET SECTION.
28500        COPY ALL SETS.
28600    END-SCHEMA.
```

9.3. *Détermination des paramètres physiques*

Par manque d'information, nous ne pouvons pas déterminer les paramètres physiques de façon complètement satisfaisante. En plus, cette détermination demanderait une étude plus approfondie, basée sur les statistiques des données ainsi que sur les caractéristiques d'un SGBD.

Notre démarche sera donc la suivante :

1. Prédéterminer une description du schéma
2. Evaluer les quantifications en nous basant sur cette description et des hypothèses et en tirer quelques conclusions
3. Examiner les applications envisagées
4. Revenir à 1 avec des modifications éventuelles

Dans le cadre de notre travail, nous faisons un calcul approximatif de la taille de cette BD en basant sur des hypothèses qui nous semblent raisonnables.

a) quantifications

Nous utilisons le mode de représentations des valeurs de 7-BIT (ASCII), ce qui rend plus compatible avec des autres systèmes

- la longueur des types de record et de leurs associations

SCHEMA-REC : 12 caractères + 4 pointeurs	= 7 words
SUB-REC : 37 caractères + 1 pointeur	= 9 words
AREA-REC : 135 caractères + 1 pointeur	= 28 words
REC-TYPE : 77 caractères + 5 pointeurs	= 21 words
SET-TYPE : 73 caractères + 4 pointeurs	= 19 words
WITH-REC : 3 caractères + 1 pointeur	= 2 words
MEMBERS : 4 caractères + 3 pointeurs	= 4 words
SOS : 3 caractères + 3 pointeurs	= 4 words
ITEM : 88 caractères + 2 pointeurs	= 20 words
USING-REC : 5 caractères + 2 pointeurs	= 3 words
VARIABLE : 35 caractères + 1 pointeur	= 8 words
ITEM-SUB : 156 caractères + 1 pointeur	= 33 words

Nous supposons que dans la base de données des schémas, il y a en moyenne :

- 6 schémas
- 15 types de record/schéma
- 20 types de set/schéma
- 3 aréas/schéma
- 5 sub-schéma/schéma
- 7 items/type de record

- 1,5 aréas/type de record : les records d'un certain type se trouvent dans 1,5 aréas (ce qui nous permet de calculer le nombre des réalisations de WITH-REC)
- 1,2 type member/type de set
- 20% de chemins d'accès basés sur type owner (SOS avec LOCATION MODE OF OWNER)
- 2 pas/chemin d'accès
- 2 items d'identifiant de set/pas d'accès
- 30% de sets déclaré avec un ordre de tri
- 3 items/clé de tri
- 40% de type de record déclaré avec LOCATION MODE DIRECT
- 25% des identifiants de set déclaré avec la clause ALIAS
- 30% des items déclarés comme items composés.

Ce qui nous donne la taille d'un schéma qui est de l'ordre de 4.447 "words".
Si une page est composée de 512 words

4447/512 = 8,6 pages

En tenant compte des "overhead", nous pouvons conclure qu'il y a (9-10) pages par schéma.
Pour 6 schémas, on a : (54-60) pages
Notons encore que ces suppositions sont tout à fait hypothétiques

Ainsi en réservant un tampon de 10 pages, on a suffisamment de place pour contenir toutes les informations d'un schéma. (les deux applications envisagées ont besoin d'une grande partie de ces informations). Ce qui permet de dire que le temps d'accès nécessaire à ces 2 applications sera dans l'hypothèse inopinée, très raisonnable.

b) Les applications

L'examen du graphe d'accès complet et les algorithmes des 3 programmes d'application ainsi que le caractère hiérarchique du DDL-DBMS-20, nous permet de conclure qu'en général on a besoin de toutes les descritpions d'une BD DBMS-20, cela entraîne qu'une organisation en bloc est convenable :

- Nous créons un seul aréa
- nous choisissons LOCATION MODE VIA pour :

SUB-REC
AREA-REC
REC-TYPE
WITH-REC
SET-TYPE
MEMBERS
SOS
ITEM
ITEM-SUB

c) conclusions générales

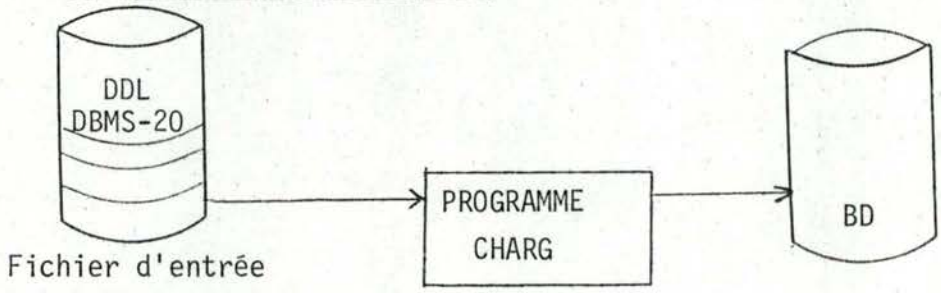
1. En raison du caractère imprévisible de la quantification de cette BD, nous devons en surestimer la charge
2. Notre étude de la performance est très sommaire, mais dans notre cas, l'exécution de 3 programmes d'application ne pose pas un grand problème de temps. En effet, après l'exécution du programme de génération des listings et de celui de générations des tables pour compilateur DML , les informations sont enregistrées dans des fichiers.
3. Lorsque la description d'un schéma DBMS-20 est trop grande, on peut encore augmenter le nombre de tampons. En effet, nous croyons qu'une quinzaine de tampons est encore acceptable dans une machine avec la taille de mémoire centrale assez grande.

° ° °

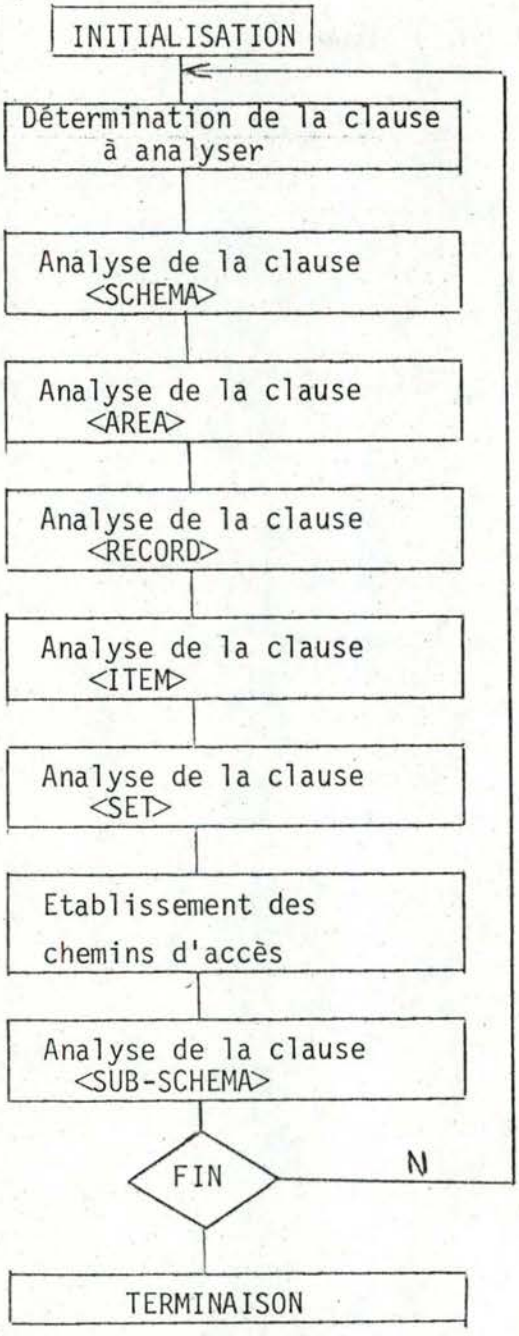
CHAPITRE X : LES PROGRAMMES

Nous décrivons dans ce chapitre les traitements principaux de 3 programmes : le programme de chargement, de générations des listings de DDL et des tables du compilateur DML du modèle DBMS-20 simplifié. Les traitements détaillés et les explications sont présentés dans un dossier séparé (voir annexe).

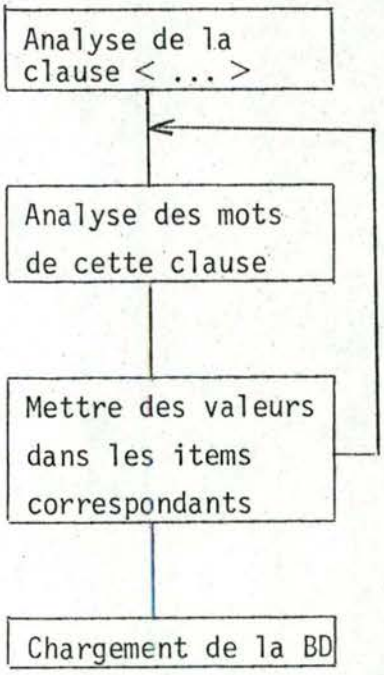
10.1. Le programme de chargement



Le fichier d'entrée contient les programmes DDL DBMS-20 (décrivant les BD DBMS-20) qu'on veut exploiter en utilisant le DML-MCS. Le programme de chargement analyse les descriptions contenues dans le fichier d'entrée et les enregistre dans la BD du schéma (décrit dans le chapitre IX)



Pour chaque clause, le programme fait des analyses comme suit :



10.2. *Programme de génération des listings DDL-MCS*

Le générateur des listings DDL du modèle simplifié (du DBMS-20) est conçu pour

- la génération de tous les sub-schémas existants dans la BD
- la génération d'un sub-schéma d'un schéma
- les générations de tous les sub-schémas d'un schéma particulier.

1ère génération

A partir de la BD ainsi construite, le générateur crée :

- le fichier DDLDIR qui est la "directory" de tous les fichiers de "listings DDL"
- les fichiers contenants les "listings DDL", chaque fichier est la description d'un sub-schéma (du modèle simplifié)

2ème génération

Le générateur met à jour le fichier DDLDIR et crée des fichiers de listing DDL

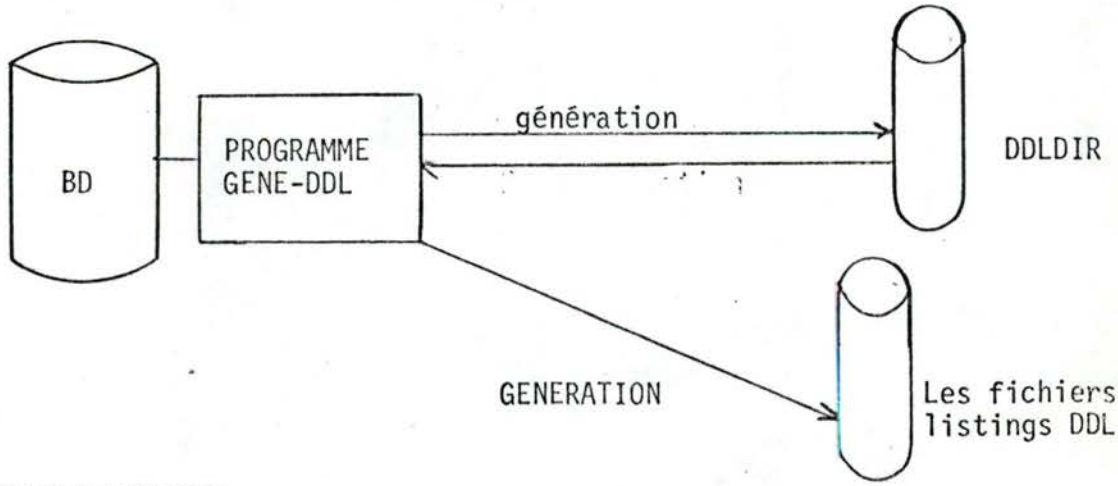
Après la génération, on a des fichiers dont les noms sont formés de la manière suivante :

<nom-schéma> <numéro-sub-schéma> <0> (*)

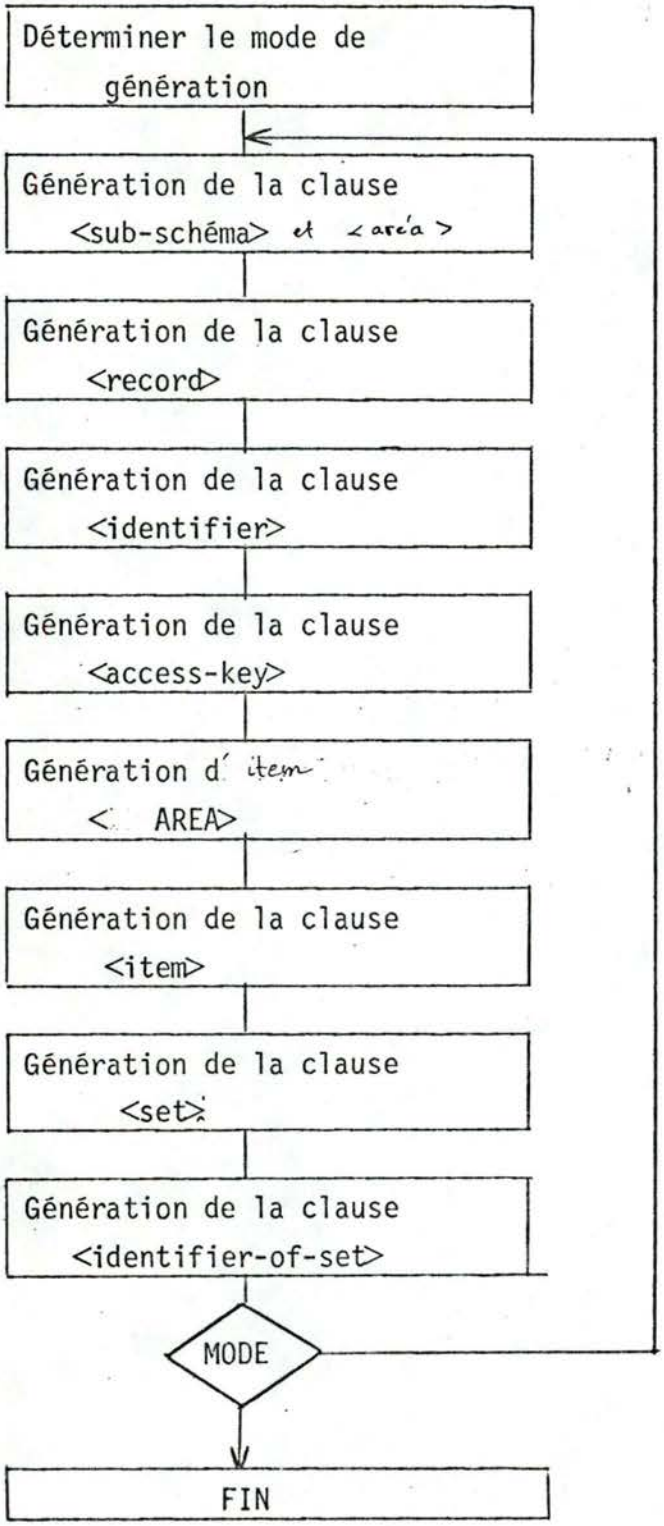
Tous ces noms de fichiers se trouvent dans un fichier glocal "directory" DDLDIR dont les articles ont la forme

<nom-schéma> <numéro-sub-schéma> <0> <nom-sub-schéma>

Les utilisateurs vont consulter ce fichier avant de faire imprimer les listings DDL



(*) pour distinguer avec les fichiers du compilateur DML


Déterminer le mode de génération
Génération de la clause <sub-schéma> et <area>
Génération de la clause <record>
Génération de la clause <identifier>
Génération de la clause <access-key>
Génération d'item <AREA>
Génération de la clause <item>
Génération de la clause <set>
Génération de la clause <identifier-of-set>
MODE
FIN

10.3. *Programme de génération des tables pour DML*

De même, il y a 3 modes de génération des tables pour compilateur DML comme dans le cas précédent.
Autre que le fichier "directory" DML, la génération des tables DML d'un sub-schéma nécessite 5 fichiers temporaires :

- F-ARS contenant les tables AREA-TAB, REC-TAB, SET-TAB
- F-ITEM contenant la table ITEM-TAB
- F-WOA contenant les tables WITHIN-TAB, OWNER-MEM-TAB et ACCESS-TAB
- F-ITEMSS contenant la table ITEM-SS-TAB
- F-VAR contenant la talbe VARIABLE-TAB

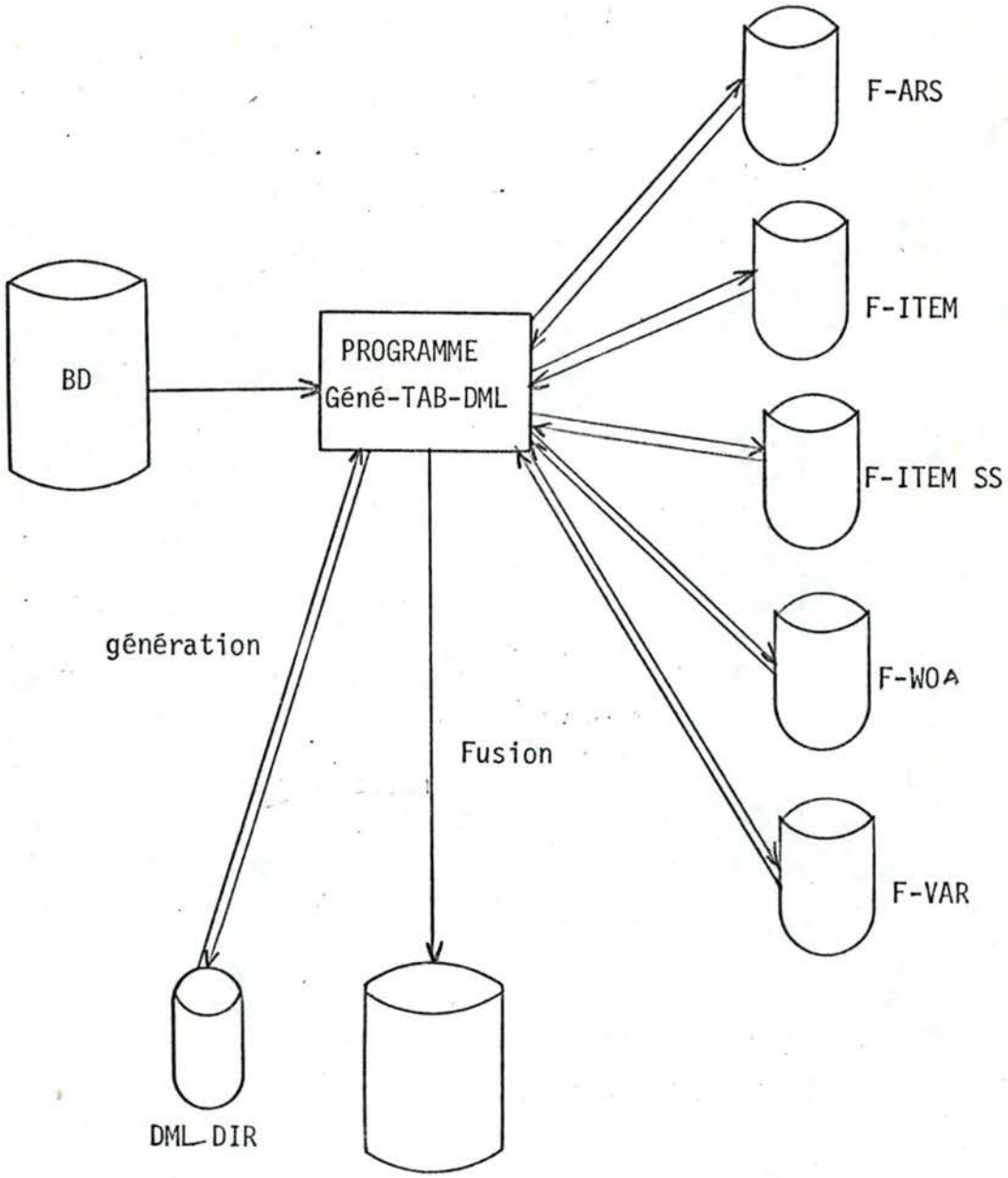
A partir de ces 5 fichiers, le générateur fait une fusion en vue de construire un seul fichier contenant toutes les tables d'un sub-schéma, nécessaires pour le compilateur DML

Après la génération, on a des fichiers dont les noms sont formés de la manière suivante :
<nom-schéma> <numéro-sub-schéma> <m> (*)

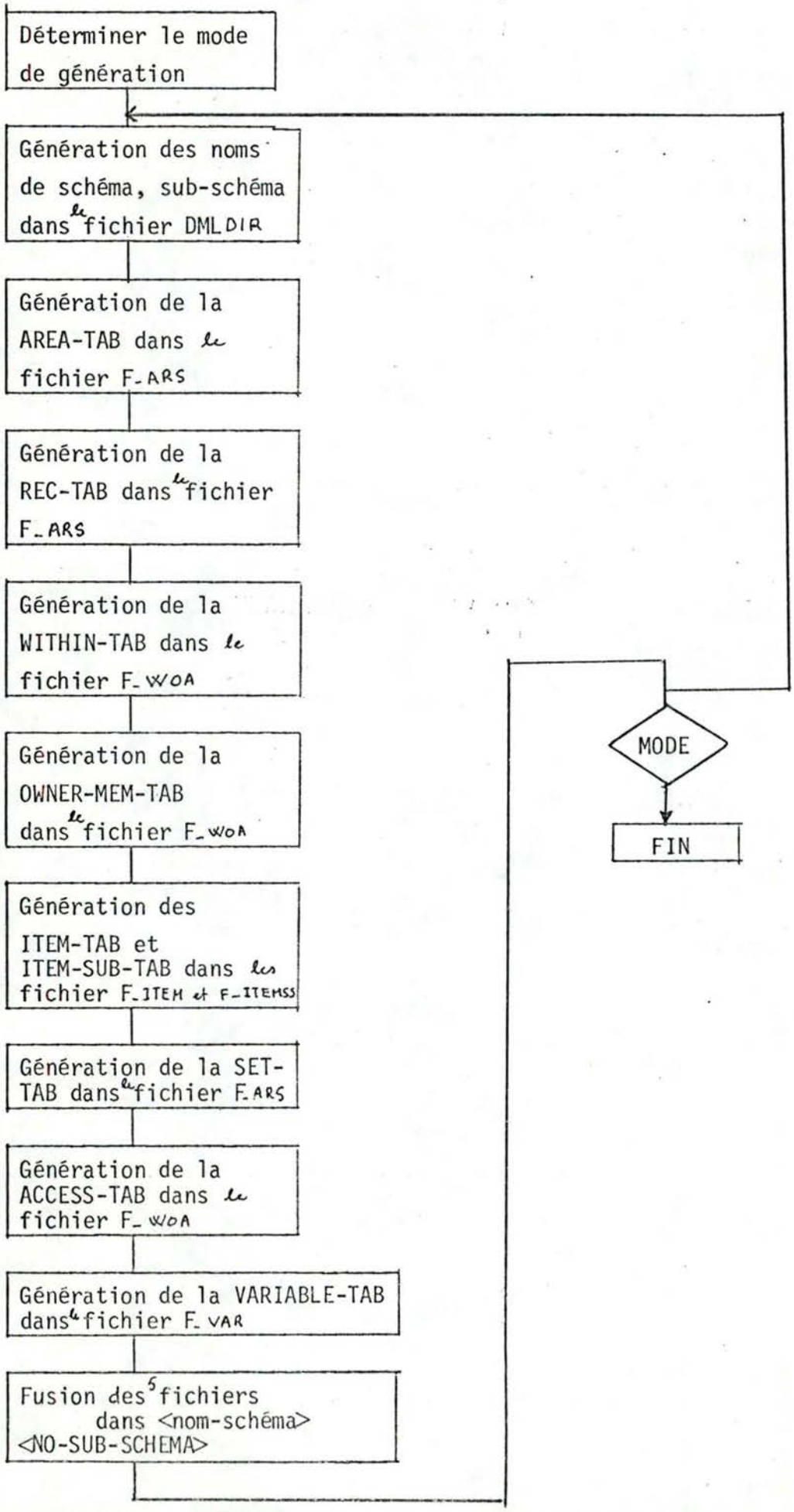
Tous ces noms de fichiers se trouvent dant un fichier "directory" DML_DIR dont les articles ont la forme
<nom-schéma> <numéro-sub-schéma><m><nom-sub-schéma>

Le compilateur DML va consulter ce fichier "directory" avant de charger les tables contenues dans le fichier correspondant.

(*) pour distinguer avec les fichiers des listings DDL



Les fichiers du DML


Déterminer le mode de génération
Génération des noms de schéma, sub-schéma dans le fichier DML DIR
Génération de la AREA-TAB dans le fichier F_ARS
Génération de la REC-TAB dans le fichier F_ARS
Génération de la WITHIN-TAB dans le fichier F_WOA
Génération de la OWNER-MEM-TAB dans le fichier F_WOA
Génération des ITEM-TAB et ITEM-SUB-TAB dans les fichier F_ITEM et F_ITEMSS
Génération de la SET-TAB dans le fichier F_ARS
Génération de la ACCESS-TAB dans le fichier F_WOA
Génération de la VARIABLE-TAB dans 4 fichier F_VAR
Fusion des 5 fichiers dans <nom-schéma> <NO-SUB-SCHEMA>
MODE
FIN

CONCLUSION

Nous pensons que ce travail permet de

- représenter au niveau externe, le modèle CODASYL par un autre modèle beaucoup plus simple
- construire un langage de manipulation de BD de haut niveau basé sur le SGBD-CODASYL

La simplicité du modèle découle essentiellement de la suppression (en fait de l'occultation) de certains mécanismes délicats à la charge du programmeur DML : LOCATION, SET OCCURRENCE SELECTION, mécanisme des CURRENT; ceux-ci sont en effet automatiquement pris en charge par le compilateur DML-MCS. La simplicité découle aussi d'un effort d'uniformisation de certains concepts : ACCESS-KEY et IDENTIFIER.

Le modèle CODASYL simplifié (construit de telle manière) permet d'exploiter les BD-CODASYL d'une façon plus facile et plus simple (grâce à un langage de haut niveau de type DML-SPHINX) : il n'exige pas une connaissance des aspects physiques d'implémentation des données.

Mais il risque d'être moins efficace à l'exploitation; en effet :

- la génération des ordres DML-CODASYL (réalisée par le compilateur DML-MCS) peut être redondante et volumineuse
- parfois les possibilités offertes par le SGBD ne sont pas exploitées, nous citons quelques cas :
- un DML de type DML-SPHINX (en général, un langage de haut niveau) ne permet pas certaines optimisation physiques (rangement dans une "page" voulue : LOCATION MODE DIRECT)
- de même, le mécanisme d'accès par un chemin prédéterminé (SOS) qui ici est commandé par le compilateur DML-MCS, ne peut être utilisé dans toute sa puissance.
- le positionnement des CURRENT est un moyen efficace d'accéder aux enregistrements, ces CURRENT sont gérés par le compilateur DML-MCS ce qui entraîne une certaine redondance.

Ces problèmes demandent une parfaite connaissance des mécanismes de gestion que nous voulons éviter.

A partir du travail que nous avons réalisé, il y a un nombre de prolongements possibles :
Notre travail correspond à la démarche inverse de l'implémentation d'un modèle, en effet en nous basant sur le modèle CODASYL (et le DML-SPHINX associé au modèle d'accès), nous en définissons un autre plus simple. Inversément, nous pensons qu'on peut implémenter un sous-ensemble du modèle d'accès en employant des spécifications CODASYL.

D'autre part, la description des structures CODASYL qui est enregistré dans la base de données est suffisamment générale pour admettre d'autres applications telles que la génération automatique des programmes d'interrogation et de mise à jour, ou encore, observant que le sub-schéma CODASYL (et en particulier DBMS-20) est un document s'adressant surtout à l'administrateur), on pourrait générer pour l'utilisateur une description plus simple et plus complet qui ne l'oblige pas à consulter le schéma.

Enfin, nous croyons qu'on peut faire rapidement ce travail avec des autres systèmes de type CODASYL en se basant sur le schéma conceptuel et le modèle simplifié.

°
° °

ANNEXE

Le fichier DBMSO.DDL contient la description du schéma

Le fichier CHARG.CBL contient le programme de chargement

Le fichier BDTEST.DDL est le fichier de tests qui contient la description d'un schéma DBMS-20 (fichier d'entrée du programme de chargement)

Le fichier DB SPHX-DBS est la BD du schéma DBMSO.DDL

Le fichier GENE-DDL.CBL contient le programme de génération de listing DDL

Le fichier GENE-TAB.CBL contient le programme de génération des tables DML

Les deux fichiers "directory" DDLDIR et DMLDIR contiennent les noms de fichiers générés par les deux derniers programmes

En plus, nous construisons le fichier COMMENT.DAT qui contient les commentaires :

- de la description du schéma (DBMSO-DDL)
- du programme de chargement (CHARG.CBL)
- du programme de génération des listings DDL (GENE-DDL.CBL)
- du programme de génération des tables DML (GENE-TAB.CBL).

BIBLIOGRAPHIE

1. SYSTEME DE CONCEPTION ET D'EXPLOITATION DE BASES DE DONNEES
   - 1ère partie : Modèles et Langages
   - 2ème partie : Manuel de référence des langages

   projet de recherche C.I.P.S. N° I.2/15
   Institut d'Informatique
   Facultés Universitaires Notre-Dame de la Paix - Namur

2. SYSTEME DE GESTION DE BANQUE DE DONNEES SPHINX

   Projet N° 2
   Institut d'Informatique
   Facultés Universitaires Notre-Dame de la Paix - Namur

3. B. LE CHARLIER et J.L. HAINAUT : Modèles, langages et système pour la conception de l'exploitation de bases de données

4. J.P. LALOUX : Implémentation du modèle d'accès par lui-même.
   Mémoire de fin d'études, Institut d'Informatique de Namur (1976)

5. CODASYL : Data Base Task Group 1971 - Report

6. C.J. DATE : An introduction to Database Systems

7. Data Structure Models for Information SYstems
   Institut d'Informatique de Namur

8. Data Base Description. IFIP
   North-Holland/American Elsevier

9. JOACHIM W. SCHMIDT : Some High level Language constructs for Data of type Relation : ACM 9-1977

10. DIGITAL DATA BASE MANAGEMENT SYSTEM
    Programmer's Procedures Manual

11. DIGITAL : DBMS
    Administrator's Procedures Manual

12. DELVAUX : Implémentation d'un langage de haut niveau de manipulation de bases de données"
    Mémoire de fin d'étude - 1977

13. HAINAUT J.L. : Etude de la Set Occurence Sélection dans les rapports Codasyl
    Publications de l'Institut d'Informatique à Namur - 1979

14. TAYLOR F. : Codasyl Data-Base Management System
    ACM Computing Survey - volume 8, number 1 - march 1976

15. FRY S. : Evolution of Data Base Management System
    ACM Computing Survey - volume 8, number 2 - march 1976

16. MICHAELS, MITTMAN, CARLSON : A comparison of relational and Codasyl approaches to Data-Base Management
    ACM Computing Survey - volume 8, number 1 - march 1976

17. HUITS : Requirements for languages in data-base systems
    ACM Douqué and g.m. Nyssen - 1975